E^2AG: Entropy-Regularized Ensemble Adaptive Graph for Industrial Soft Sensor Modeling

Zhichao Chen, Licheng Pan, Yiran Ma, Zeyu Yang *Member, IEEE*, Le Yao *Member, IEEE*, Jinchuan Qian *Member, IEEE*, and Zhihuan Song

***Abstract*—Adaptive Graph Neural Networks (AGNNs) have achieved remarkable success in industrial process soft sensing by incorporating explicit features that delineate the relationships between process variables. This article introduces a novel GNN framework, termed Entropy-Regularized Ensemble Adaptive Graph (E^2AG), aimed at enhancing the predictive accuracy of AGNNs. Specifically, this work pioneers a novel AGNN learning approach based on mirror descent, which is central to ensuring the efficiency of the training procedure and consequently guarantees that the learned graph naturally adheres to the row-normalization requirement intrinsic to the message-passing of GNNs. Subsequently, motivated by multi-head self-attention mechanism, the training of ensembled AGNNs is rigorously examined within this framework, incorporating an entropy regularization term in the learning objective to ensure the diversity of the learned graph. After that, the architecture and training algorithm of the model are then concisely summarized. Finally, to ascertain the efficacy of the proposed E^2AG model, extensive experiments are conducted on real-world industrial datasets. The evaluation focuses on prediction accuracy, model efficacy, and sensitivity analysis, demonstrating the superiority of E^2AG in industrial soft sensing applications.**

***Index Terms*—Deep Learning, Soft Sensor, Graph Neural Networks, Mirror Descent, Reproduced Kernel Hilbert Space**

I. Introduction

PREDICTING hard-to-measure quality variables from easy-to-obtain, high-dimensional sensory data is essential for advancing intelligent manufacturing [1], aiding decision-making [2], and improving anomaly detection & diagnostic accuracy [3–5] in the industrial sector. To this end, researchers have explored a variety of approaches to develop this 'soft sensor', incorporating both mechanistic equations and data-driven models to overcome these challenges. While mechanistic equations have traditionally provided valuable insights into the complexities of industrial processes, the rapid pace of technological advancement has made them increasingly cumbersome to formulate and apply. As a result, the industry has gravitated toward more accessible, data-driven models, which are prized for their ease of use and reduced dependency on extensive prior knowledge.

This shift reflects a broader trend toward leveraging computational power and artificial intelligence to address complex industrial challenges. Amidst these advancements, deep learning (DL) and, more specifically, neural network-based (NN-based) approaches have gained prominence within the realm of industrial soft sensing. These neural networks can be broadly classified into several types, each suited for specific tasks: (stacked-) auto-encoders [6] streamline dimensional reduction, convolutional neural networks (CNNs) [7, 8] excel in spatial data analysis, recurrent neural networks (RNNs) [9] adeptly handle sequential information, and Transformers [10, 11] offer robust performance in tasks requiring attention mechanisms. Notably, the efficacy of these NN architectures is largely attributed to their rigorously designed feature extraction modules, which are pivotal for their success in various applications. For instance, CNNs leverage spatial hierarchies to extract pertinent features from images effectively, making them indispensable in visual data analysis. Similarly, RNNs, with their unique ability to process data sequentially, are well-suited for time-series analysis, capturing dynamic temporal behaviors.

However, in the realm of industrial process monitoring, soft sensor modeling faces challenges due to complex spatial coupling relations, which can significantly affect prediction accuracy. To address these complexities, researchers are increasingly adopting graph neural network-based (GNN-based) approaches [12] that allow for explicit modeling of spatial correlations. For instance, Jia *et al*. [13] employed a multi-layer graph convolution network (GCN) structure to enhance the penicillin fermentation process's soft sensor capabilities, achieving significant improvements in predictive accuracy and insight into the process dynamics. Similarly, Wang *et al*. [14] integrated a graph convolution network with a multi-head graph attention network for the anaerobic digestion process, which led to notable enhancements in prediction performance. It is important to note the inherent challenge in directly deriving a graph from complex industrial processes, given their intricate mass, heat, and information flow dynamics. Recent efforts in this domain focus on developing graphs with

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62473103, 62203169, and 62473121. This work was also suppoerted in part by the Postdoctoral Science Foundation of Zhejiang Province under Grant ZJ2023011 (Corresponding author: Jinchuan Qian and Zhihuan Song).

Zhichao Chen, Licheng Pan, Yiran Ma, and Jinchuan Qian are with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: 12032042@zju.edu.cn; 22132045@zju.edu.cn; mayiran@zju.edu.cn; qianjinchuan@zju.edu.cn).

Zhihuan Song is with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China. He is also with Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China (songzhihuan@zju.edu.cn).

Zeyu Yang is with the Huzhou Key Laboratory of Intelligent Sensing and Optimal Control for Industrial Systems, School of Engineering, Huzhou University, Huzhou 313000, China (e-mail: yangzeyu@zjhu.edu.cn).

Le Yao is with the School of Mathematics, Hangzhou 311121, Hangzhou Normal University. He is also with Qinting Data & Intelligence Co., Ltd., Hangzhou 311121, China (email: yaole@hznu.edu.cn).

learnable structures to construct more adaptive GNN-based soft sensors. For example, Wang *et al*. [15] innovatively treated the graph structure as a learnable parameter within the stacked autoencoder (SAE) model, proposing a novel stacked GCN approach that leverages this adaptability for improved sensor accuracy. Moreover, Zhu *et al*. [16] advanced this concept by combining a multi-hop attention mechanism with multiple learnable graph structures, thus boosting feature extraction capabilities and validating their method's effectiveness in a coal mill rig process. These enhancements underscore the evolving landscape of industrial soft sensor modeling, where the integration of advanced NN architectures and the explicit modeling of spatial relationships are pivotal for advancing the field's capabilities and applications.

Even though these works have proven the concept of the adaptive graph learning strategy, there are two long-ignored issues remaining to be addressed:

1) **Row Normalization Constraint Satisfaction:** To the best of our knowledge, current gradient descent-based DL backends promise the learned graph satisfies the row normalization constraint inherent from message-passing mechanism in GNNs, where the row-sum of the learned graph should be equal to 1. To alleviate this issue, current AGNNs design the 'Softmax' operator during model inference stage. The question of how to satisfy the row normalization constraint during the training stage has not yet been answered.
2) **Highly Efficient Ensemble Mechanism:** While the analogy between Transformers and GNNs has been established, demonstrating the success of graph ensembling akin to the multi-head self-attention mechanism in Transformers, such ensembling could introduce parameter redundancy and not necessarily lead to clear performance enhancements. Identifying methods to ensure the efficiency of GNN module ensembles with diversity guarantee, thereby enhancing performance without accruing redundancy, remains an open question for research.

The key to addressing issue 1) is treating the graph learning as a constrained optimization problem where the learned graph should satisfy the row-sum equal to 1, and derive the corresponding iteration equations compatible with gradient descent-based DL backends represented by PyTorch [17] and JAX [18]. Based on the analysis results of issue 1), the key to addressing issue 2) is analyzing the self-attention mechanism and designing corresponding regularization function in the learning objective for the sake of improving the diversity of the learned graph.

To achieve these objectives, this work initially reformulates graph learning within the neural architecture as a constrained optimization problem. It is then addressed within the mirror descent framework, leading to the derivation of corresponding iterative equations. Subsequently, based on the analytical results, the ensemble mechanism of the multi-head self-attention mechanism (MHSAM) is further examined. It is reformulated as a functional optimization problem concerning the expectation function from a Dirichlet distribution. Entropy regularization is introduced to foster diversification of this Dirichlet distribution, and its closed-form iterative equation is derived by constraining the functional derivative in reproduced kernel Hilbert space (RKHS). Ultimately, the proposed model, named Entropy-Regularized Ensemble Adaptive Graph (E^2AG), is proposed. Its training and testing algorithms are summarized, and its effectiveness is demonstrated in two real industrial scenarios.

The contributions of this paper are listed as follows:

1) This work introduces a novel training procedure for AGNN learning, which is compatible with DL backends and inherently adheres to the row normalization constraint required by the message-passing mechanism.
2) A novel AGNN, named E^2AG, is proposed and innovated by MHSAM, featuring an ensemble mechanism and an entropy regularization term to enhance prediction learning accuracy in soft sensor modeling. Additionally, the training algorithm is rigorously derived to facilitate model training algorithm implementation.
3) Building on the innovations above, this work also offers a theoretical analysis of the learning algorithm for E^2AG, from the perspective of convergence, uniqueness, and stability.
4) Various experiments on real industrial processes are conducted to validate the efficacy of the proposed E^2AG.

The rest of this paper is organized as follows: to better understand the technical gap and concerning background knowledge, related works, and preliminaries are summarized in Section II and Section III, respectively. On this basis, the model derivation and effectiveness validation are proposed in Section IV and Section V, respectively. Finally, the conclusions and future research directions are given in Section VI.

II. Related Works

Soft sensor modeling faces the challenge of addressing the graph-structured nature of industrial process variables [16], which is rooted in complex spatial coupling relationships. To address this challenge, prior studies have delved into GNN-based approaches to capture these relationships, with the goal of enhancing the performance of inferential sensors. However, in contrast to domains such as recommender systems [19–21], causal inference [22, 23], and Quality-of-Service data analysis [24], where the graph structure is readily available, GNN-based soft sensor modeling often requires the construction of graphs. In this regard, previous works have attempted to construct graphs based on data similarity measures like mutual information [25] and Laplacian matrices [26] between process variables. However, decoupling graph construction from model training may limit model performance. Consequently, AGNN architectures, in which the graph is learned as a model parameter, have gained traction in soft sensor modeling, similarly to GNN-based time-series forecasting [27, 28]. Utilizing learnable similarity measurements, Wang *et al*. [15] developed graphs to support multi-layer GNNs, where all data share a single graph, as innovated by Graph WaveNet [27]. Notably, modeling the graph in a data-centric paradigm is an alternative to improving the expressiveness of the model. To this end, some researchers [16, 29–31] have explored constructing graphs based on node similarity, employing self-attention

mechanisms to generate data-wise graphs. For instance, in the work of Wang *et al*. [30], the self-attention mechanism is strategically designed to construct graphs both spatially and temporally.

While previous works have achieved considerable success in AGNN-based soft sensor modeling with notable prediction accuracy across various industrial processes, there remain several unresolved issues that motivate this paper:

1) **Row Normalization Constraint Satisfaction:** Gradient descent-based DL backends cannot guarantee that the learned graph satisfies the row normalization constraint. Consequently, previous works primarily employ the Softmax operator during the model inference stage. The challenge of devising a novel algorithm that naturally satisfies the row normalization constraint while being compatible with DL backends remains unaddressed.
2) **Highly Efficient Ensembling Mechanism:** Inspired by the multi-head self-attention mechanism, previous works have attempted to ensemble multiple learnable graphs to obtain the predicted value. However, this strategy may lead to issues of parameter redundancy. Therefore, designing suitable regularization to enhance modeling efficiency poses a significant challenge.

The rest of this paper will focus on solving the abovementioned technical gaps.

III. Preliminaries

A. Message Passing Mechanism & GNNs

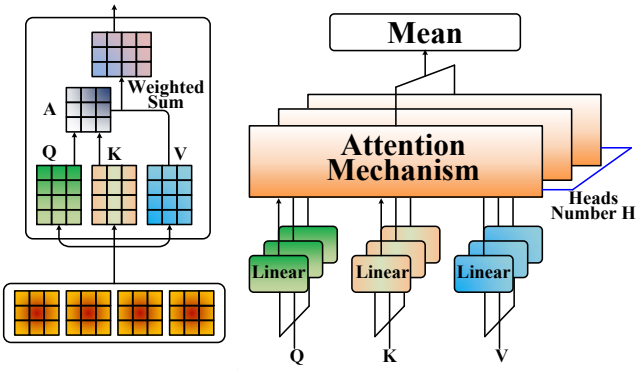
The Message Passing Neural Network (MPNN) serves as a comprehensive framework for GNNs. Several notable GNNs, such as the Graph Convolutional Network (GCN) [32, 33], GraphSAGE [34], and Graph Attention Network (GAT) [35], can be viewed as specific instances of MPNN, as detailed in reference [36]. In the MPNN framework, the feature of a node ν_i (denoted by F_i) is updated through a two-step process, as described by Equations (1) and (2): The feature of node ν_i can be updated by (1) and (2) in MPNN.

$$m_i = \sum_{\nu_j \in \mathcal{D}(\nu_i)} \text{Mess}(F_i, F_j, e_{(\nu_i,\nu_j)}), \tag{1}$$

$$F_i^{'} = \text{Update}(F_i, m_i), \tag{2}$$

where $\mathcal{D}(\nu_i)$ is the set of adjacent nodes of ν_i, Mess represents the message function, e denotes the edge feature, and Update signifies the update function. By altering the summation operator, message function, and update function, MPNN can be adapted to other GNNs for extracting spatial features of process variables. A critical aspect of the model is the row-normalization of the graph matrix $\mathcal{A}$ during inference. For example, in GCN, given a matrix $\tilde{\mathcal{A}}$ as $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$, the diagonal matrix $D = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ is computed, leading to the row-normalized matrix $\mathcal{A} = D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$. Previous AGNNs, such as Stacked-GWave [15] and DGDL [16], have employed the Softmax operator to ensure row normalization during the model inference stage.

B. Multi-Head Self Attention Mechanism



(a) (b)

Fig. 1. The illustrations of (a) SAM and (b) MHSAM.

Fig. 1 presents illustrations of the Self-Attention Mechanism (SAM) and the MHSAM [11], respectively. The computational procedure can be delineated as follows: Given an input $\mathbf{X} \in \mathbb{R}^{m\times n}$ for the Multi-Head Attention mechanism, where m represents the sequence length and n represents the data dimension, the 'query' ($\mathbf{Q} \in \mathbb{R}^{m\times d}$, d indicates the attention dimension), 'key' ($\mathbf{K} \in \mathbb{R}^{m\times d}$), and 'value' ($\mathbf{V} \in \mathbb{R}^{m\times d}$) matrices are initially computed by a linear layer as follows:

$$\begin{cases} \mathbf{Q} & = \mathbf{X}W_{\mathbf{Q}} + b_{\mathbf{Q}} \\ \mathbf{K} & = \mathbf{X}W_{\mathbf{K}} + b_{\mathbf{K}} \\ \mathbf{V} & = \mathbf{X}W_{\mathbf{V}} + b_{\mathbf{V}} \end{cases}, \tag{3}$$

Subsequently, the feature is obtained using the following equation:

$$\hat{A} = \text{Softmax}\left(\frac{\mathbf{Q}\cdot\mathbf{K}^{\top}}{\sqrt{d}}\right)\mathbf{V}, \tag{4}$$

where $\text{Softmax}\left(\frac{\mathbf{Q}\cdot\mathbf{K}^{\top}}{\sqrt{d}}\right)$ is referred to as the attention score and is denoted by A in this paper. To enhance model performance, MHSAM adopts an ensemble strategy, where Equations (3) and (4) are repeated for H times, and the mean value of these H features is outputted.

C. Mirror Descent & Bregman Divergence

Denote the loss function and parameter as $\mathcal{L}$ and w respectively, the iteration procedure of gradient descent with learning rate η between times t and $t+1$ can be reformulated as follows:

$$\begin{aligned} &w^{t+1} = w^t - \eta\nabla_w\mathcal{L}|_{w=w^t} \Rightarrow \\ &w^{t+1} = \arg\min_{w\in\Omega}[(\nabla_w\mathcal{L}|_{w=w^t})^{\top}w + \frac{1}{2\eta}\left\|w - w^t\right\|_2^2], \end{aligned} \tag{5}$$

where $\Omega \subset \mathbb{R}^d$ is constraint space. From the perspective of mirror descent, the L_2 norm $\|\cdot\|_2$ in (5) is selected as Bregman divergence $\mathbb{B}$ that reflects the geometry of a problem:

$$w^{t+1} = \arg\min_{w\in\Omega}\{(\nabla_w\mathcal{L}|_{w=w^t})^{\top}w + \frac{1}{\eta}\mathbb{B}\left[w\|w^t\right]\}, \tag{6}$$

and $\mathbb{B}$ is induced by a strongly convex, essentially smooth function $\psi : \Omega \to \mathbb{R}^{\mathrm{d}} \cup \{\infty\}$ as follows:

$$\mathbb{B}[w\|w'] = \psi(w) - \psi(w') - \nabla\psi(w')^{\top}(w - w'). \quad (7)$$

Notably, the Bregman divergence is L_2 norm when $\psi(w) = \frac{1}{2}\|w\|_2^2$.

Building upon this foundation, the solution to (6) named mirror gradient descent [37] can be derived as below:

$$w^{t+1} = \nabla\psi^*[\nabla\psi(w^t) - \eta\nabla f(w^t)], \quad (8)$$

where ψ^* is the convex conjugate of ψ, defined as $\psi^* \triangleq \sup_{w\in\Omega}[\xi^{\top}w - \psi(\theta)]$, and $\nabla\psi$ acts as a bijective mapping from the domain Ω to the domain $\mathrm{dom}(\psi^*)$ (with the variable denoted by ξ), possessing an inverse mapping $(\nabla\psi)^{-1} = \nabla\psi^*$. Consequently, (8) can be interpreted in the following manner [38]: initially, w^t is mapped to ξ^t via $\nabla\psi$, followed by the application of gradient descent with a learning rate η to obtain $\xi^{t+1} = \xi^t - \eta\nabla_w\mathcal{L}|_{w=w^t}$, and ultimately, this result is mapped back to w^{t+1} using $\nabla\psi^*(\xi^{t+1})$.

IV. Proposed Approach

A. Problem Statement & Notifications

This work is primarily concerned with the supervised industrial soft sensor task. For this purpose, a set of past labeled samples $\{(x_i, y_i)\}|_{i=1}^{\mathrm{N_{train}}}$ is given, where $x_i \in \mathbb{R}^{\mathrm{M}}$ represents the easy-to-measure process variables (also known as covariates), $y_i \in \mathbb{R}^1$ represents the hard-to-measure quality variable (also known as labels), and $\mathrm{N_{train}}$ denotes the size of the training dataset.

The objective is to train a GNN that takes covariates x as input, employs a *row-normalized graph* $\mathcal{A}$ as a learnable model parameter, and outputs the quality variable y. It is noteworthy that the graph $\mathcal{A}$ is a square matrix of size $\mathrm{M}\times\mathrm{M}$. For a clearer elucidation of the derivation process, each row of graph $\mathcal{A}$ is denoted by vector $\vec{a}_i|_{i=1}^{\mathrm{M}}$, allowing $\mathcal{A}$ to be decomposed as $\mathcal{A} = [\vec{a}_1^{\top}, \ldots, \vec{a}_{\mathrm{M}}^{\top}]^{\top}$, where $\mathcal{A}_{i,j}$ represents the entity in the i-th row and j-th column.

B. Satisfaction of Row Normalization Constraint

First, consider the iterative process of graph $\mathcal{A}$:

$$\min_{\mathcal{A}} \quad \mathcal{L}(y, \mathcal{A}, x) \quad \text{s.t.} \quad \begin{cases} \sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} = 1 \\ \mathcal{A}_{i,j} \geq 0 \end{cases}. \quad (9)$$

It should be noted that directly solving this optimization problem using standard gradient descent, as defined in (10) with a learning rate η, is infeasible due to its inability to satisfy the row normalization constraint.

$$\mathcal{A}^{t+1} = \mathcal{A}^t - \eta \times \nabla_{\mathcal{A}}\mathcal{L}(y, \mathcal{A}, x)|_{\mathcal{A}=\mathcal{A}^t}. \quad (10)$$

This limitation can be understood as follows: the Euclidean norm is not an appropriate discrepancy metric for the multivariate variable $\mathcal{A}_{i,j}|_{i=1}^{\mathrm{M}}$, whose summation is constrained to be 1 (i.e., $\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} = 1$). In other words, even if each row of $\mathcal{A}^t$ satisfies the row normalization constraint, the updated $\mathcal{A}^{t+1}$ obtained through (10) may not necessarily fulfill the same constraint.

Meanwhile, note that the summation equal to 1 can be treated as a special kind of probability distribution. Consequently, based on references [38, 39] the entropy function can be adopted as $\psi(\cdot)$ function for this scenario:

$$\psi(\mathcal{A}_{i,j}) = \sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j}\log(\mathcal{A}_{i,j}) - \mathcal{A}_{i,j}. \quad (11)$$

On this basis, the following proposition is given to promise the $\mathcal{A}$ satisfies the simplex constraint during iterative procedure within gradient descent framework based on the mirror descent in Section III-C:

Proposition 1. *The row vector $\vec{a}_i|_{i=1}^{\mathrm{M}}$ can be optimized via the following equations:*

$$\begin{cases} \log\vec{a}_i^{t+1} = \log\vec{a}_i^t - \eta\nabla_{\vec{a}_i}\mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t} + \lambda_i \\ e^{-\lambda_i} = \left\{\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j}\exp\{-\eta[\nabla_{\vec{a}_i}\mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t}]_{i,j}\}\right\} \end{cases}. \quad (12)$$

Proof. Plugging (11) into (6), the following learning objective is obtained:

$$[\nabla_{\vec{a}_i}\mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t}]^{\top}\vec{a}_i + \frac{1}{\eta}\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j}\log(\frac{\mathcal{A}_{i,j}}{\mathcal{A}_{i,j}^t}) - \mathcal{A}_{i,j} + \mathcal{A}_{i,j}^t. \quad (13)$$

To promise $\mathcal{A}_{i,j} \geq 0$ holds, the celebrated Lagrangian multipliers $\lambda_i \in \mathbb{R}$ and $\vec{\mu}_i \in \mathbb{R}^{\mathrm{M}}$ are introduced to handle the equality constraint $\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} = 1$ and inequality constraint $\mathcal{A}_{i,j} \geq 0$. And thus, (13) can be reformulated as follows:

$$\begin{aligned} \mathcal{L}^u(y, \mathcal{A}, x) \triangleq\ & [\nabla_{\vec{a}_i}\mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t}]^{\top}\vec{a}_i \\ & + \frac{1}{\eta}\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j}\log(\frac{\mathcal{A}_{i,j}}{\mathcal{A}_{i,j}^t}) - \mathcal{A}_{i,j} + \mathcal{A}_{i,j}^t \\ & - \lambda(\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} - 1) + \vec{\mu}_i^{\top}\vec{a}_i, \end{aligned} \quad (14)$$

where the superscript u indicates the 'unconstrainted'. On ths basis, applying $\nabla_{\vec{a}_i}\mathcal{L}^u(y, \mathcal{A}, x) = 0$ to (14), the following equation is obtained:

$$\log\vec{a}_i = \log\vec{a}_i^t - \eta\nabla_{\vec{a}_i}\mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t} + \lambda_i - \vec{\mu}_i. \quad (15)$$

Consequently, the solution can be further reformulated as follows:

$$\vec{a}_i = \exp(\lambda_i)\exp(\log\vec{a}_i^t - \eta\nabla_{\vec{a}_i}\mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t} - \vec{\mu}_i), \quad (16)$$

where $\vec{a}_i \geq 0$ since $\exp(\cdot) \geq 0$ holds. Meanwhile, for constrained optimization problems, the following Karush-Kuhn-Tucker conditions should be satisfied:

$$\begin{cases} \mu_{i,j} \geq 0 \\ \vec{\mu}_i^{\top}\vec{a}_i = 0 \\ \sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} = 1 \\ \nabla_{\vec{a}_i}\mathcal{L}^u(y, \mathcal{A}, x) = \mathbf{0} \end{cases}. \quad (17)$$

Consequently, based on (16), the inequality constraint is inactive. Consequently, $\vec{\mu}_i = \mathbf{0}$, and λ_i is obtained based on $\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} = 1$:

$$\begin{aligned}&\exp(-\lambda)\\ =&\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} \exp\{-\eta[\nabla_{\vec{a}_i} \mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t}]_{i,j}\}. \end{aligned} \tag{18}$$

□

Upon examining (12), it becomes evident that the learning process of $\mathcal{A}$ is divided into two distinct stages. For clarity, the first stage, denoted as $\log \vec{a}_i^{t+1} = \log \vec{a}_i^t - \eta \nabla_{\vec{a}_i} \mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t} + \lambda_i$, is referred to as the gradient descent stage. This stage is analogous to the conventional gradient descent used in network training and can be readily implemented with DL backends. The second stage, expressed as $e^{-\lambda_i} = \{\sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} \exp\{-\eta[\nabla_{\vec{a}_i} \mathcal{L}(y, \mathcal{A}, x)|_{\vec{a}_i=\vec{a}_i^t}]_{i,j}\}\}$is termed the normalization stage, emphasizing the adjustment for row normalization of the matrix $\mathcal{A}$. Up to now, the iterative procedure for AGNNs for row normalization constraint satisfaction is established.

C. Model Ensembling & Entropy Regularization

1) An Expectation View of Ensemble: Based on Section III-B, the loss function for ensembling multiple graph can be given as follows based on the MHSAM:

$$\mathcal{L}(y, \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x), x), \tag{19}$$

where $f_\theta(\cdot)$ represents the downstream NN structure with inputs graph $\mathcal{A}_h|_{h=1}^{\mathrm{H}}$ and covariate x. Notably, the term $\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x)$ approximates the following integral:

$$\int \rho(\mathcal{A}) f_\theta(\mathcal{A}, x) \mathrm{d}\mathcal{A} \approx \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x). \tag{20}$$

The left-hand side of (20) can be interpreted as computing the expectation of the function $f_\theta(\mathcal{A}, x)$ with respect to the density function $\rho(\mathcal{A})$. On this basis, *the cornerstone of enhancing ensemble efficiency lies in determining the distribution* $\rho(\mathcal{A})$ such that the sampled graphs $\mathcal{A}$ are as diverse as possible.

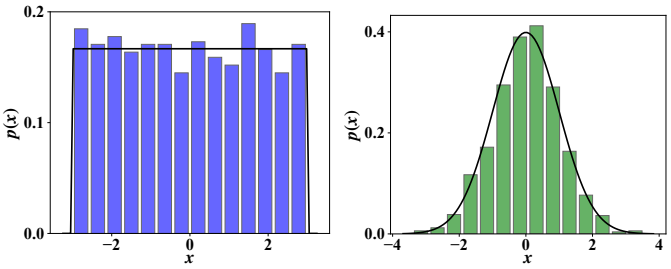
Building upon this foundation, we consider two distributions as illustrated in Fig. 2. The samples drawn from a uniform distribution $\mathcal{U}[-3, 3]$ (depicted in Fig. 2 (a), with an entropy of $\log[3-(-3)] \approx 1.79$) are spread out along the time axis. In contrast, samples from a normal distribution $\mathcal{N}(0, 1)$ (shown in Fig. 2 (b), with an entropy of $\frac{1}{2}\log(2\pi \times e) \approx 1.42$) are clustered around 0. Evidently, the former, with its higher entropy, represents the desired form, characterized by greater dispersion. In light of this, to quantify the diversification of a distribution, the following entropy function is introduced:

$$\mathbb{H}(\rho) = -\int \rho(\mathcal{A}) \log \rho(\mathcal{A}) \mathrm{d}\mathcal{A} \approx -\frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} \log \rho(\mathcal{A}_h). \tag{21}$$

And thus, the entropy regularized loss function can be given as follows:

$$\begin{aligned}\mathcal{L}_{\mathrm{ER}}(y, \mathcal{A}, x) &\triangleq \mathcal{L}(y, \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x), x) - \varphi \mathbb{H}(\rho)\\ &\approx \mathcal{L}(y, \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x), x) + \varphi \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} \log \rho(\mathcal{A}), \end{aligned} \tag{22}$$

where the negative entropy regularization is added to encourage the diversification of the learned graphs, φ is the entropy regularization strength, and 'ER' is the abbreviation of entropy regularization. For simplicity, the rest of this manuscript is organized by default with $\varphi = 1.0$ unless stated otherwise.



(a) Histogram of $\mathcal{U}[-3, 3]$ (b) Histogram of $\mathcal{N}(0, 1)$

Fig. 2. The histograms of different distributions.

2) Closed Form Optimization Equation within RKHS: Note that, (22) is a functional optimization problem. To this end, first consider the functional derivative by introducing a small perturbation $\eta\phi(\mathcal{A})$ according to references [40–42]:

$$\begin{aligned}&\mathcal{L}_{\mathrm{ER}}(y, \mathcal{A} + \eta\phi(\mathcal{A}), x) - \mathcal{L}_{\mathrm{ER}}(y, \mathcal{A}, x)\\ =&\mathcal{L}(y, \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h + \eta\phi(\mathcal{A}_h), x), x) + \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} \log \rho(\mathcal{A})\\ &-\mathcal{L}(y, \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x), x) - \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} \log \rho(\mathcal{A} + \eta\phi(\mathcal{A}))\\ \overset{\text{(i)}}{=}&\mathbb{E}_{\rho(\mathcal{A})}[\eta \nabla_{\mathcal{A}} \mathcal{L}(y, \mathcal{A}, x)^\top \phi(\mathcal{A}) - \log\det(|I + \eta\phi(\mathcal{A})|)], \end{aligned} \tag{23}$$

where the detailed derivation of equality (i) is given in Sections S.I.A and S.I.B of supplementary material, and as such the functional derivative can be given as follows:

$$\begin{aligned}&\frac{\mathrm{d}[\mathcal{L}_{\mathrm{ER}}(y, \mathcal{A} + \eta\phi(\mathcal{A}), x) - \mathcal{L}_{\mathrm{ER}}(y, \mathcal{A}, x)]}{\mathrm{d}\eta}|_{\eta=0}\\ \overset{\text{(ii)}}{=}&\mathbb{E}_{\rho(\mathcal{A})}[\nabla_{\mathcal{A}} \mathcal{L}(y, \mathcal{A}, x)^\top \phi(\mathcal{A}) - \nabla_{\mathcal{A}} \phi(\mathcal{A})]. \end{aligned} \tag{24}$$

The detailed derivation of (ii) is given in Section S.I.C of supplementary material. Based on this expression, the following proposition is introduced to obtain the closed form of (24):

Proposition 2. *Assume that the perturbation direction* $\phi(\mathcal{A})$ *is restricted in RKHS with kernel function* $\mathcal{K}(x, y)$*, (24) can be reformulated as follows:*

$$\mathbb{E}_{\rho(\mathcal{A})}[-\nabla_{\mathcal{A}} \mathcal{L}(y, \mathcal{A}, x)^\top \mathcal{K}(\mathcal{A}, \cdot) + \nabla_{\mathcal{A}} \mathcal{K}(\mathcal{A}, \cdot)]. \tag{25}$$

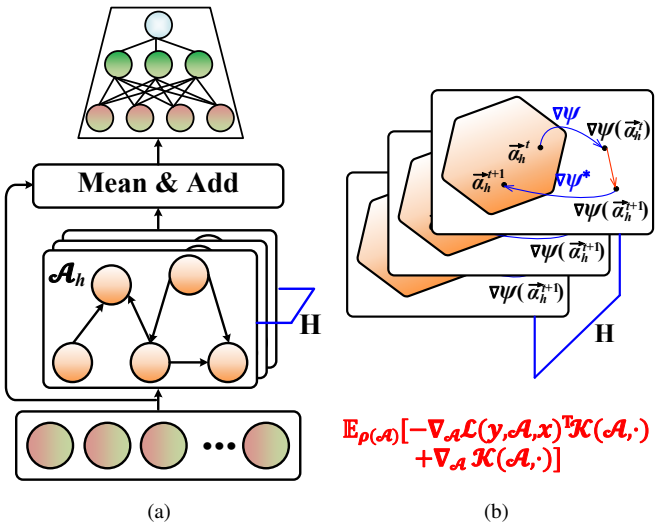
Proof. The proof is given in Section S.I.D of supplementary material. □

Substituting the gradient descent stage of (12) by (25), the iterative equations for learning E^{2}AG can be summarized by the following equations:

$$\begin{cases} \log \vec{a}_i^{t+1} = \log \vec{a}_i^t + \lambda_i - \eta \\ \times \{\mathbb{E}_{\rho(\mathcal{A})}[\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^\top \mathcal{K}(\mathcal{A},\cdot) - \nabla_{\mathcal{A}}\mathcal{K}(\mathcal{A},\cdot)]|_{\mathcal{A}=\mathcal{A}^t}\}_i \\ e^{-\lambda_i} = \sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j} \exp\{-\eta[\mathbb{E}_{\rho(\mathcal{A})}[\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^\top \mathcal{K}(\mathcal{A},\cdot) \\ \quad - \nabla_{\mathcal{A}}\mathcal{K}(\mathcal{A},\cdot)]|_{\mathcal{A}=\mathcal{A}^t}]_{i,j}\} \end{cases} \tag{26}$$

Up to now, the graph learning procedure for E^{2}AG is established. Since the learning procedure highly relies on the RKHS and mirror descent, this algorithm is named Kernelized Mirror Descent-based iteration over Simplex (KEMS) for simplicity.

D. Model Architecture



(a) (b)

Fig. 3. The illustration of (a) E^{2}AG model, (b) graph learning procedure of E^{2}AG for H = 3, and the hexagon indicates the row normalization constraint.

Based on Sections III-A, III-B and IV-C, the model architecture and the graph learning procedure are given in Fig. 3 (a) and (b), respectively. Since the ensemble strategy is designed during inference stage and the entropy regularization term is added to the learning objective, the model is named 'Entropy-Regularized Ensemble Adaptive Graph' (E^{2}AG). From Fig. 3 (a), the inference procedure of E^{2}AG model can be illustrated as follows: Let us first denote the input vector as $x \in \mathbb{R}^{1\times \mathrm{N}_{\mathrm{node}}}$. On this basis, in accordance with the methodologies detailed in Sections III-A and III-B, each dimension of the covariates is subjected to an initial transformation through an embedding layer corresponding to each head $h \in \{1,\ldots,\mathrm{H}\}$, expressed as:

$$g_{i,h}^{\mathrm{embed}} = x_i \times W_{i,h}^{\mathrm{embed}} + b_{i,h}^{\mathrm{embed}}|_{i=1}^{\mathrm{N}_{\mathrm{node}}}, \tag{27}$$

where embedding feature $g_{i,h}^{\mathrm{embed}} \in \mathbb{R}^{1\times \mathrm{N}_{\mathrm{node}}}$. After that, the message is passed through the graph matrix $\hat{\mathcal{A}}_h|_{h=1}^{\mathrm{H}}$ as the lower middle part of Fig. 3 (a) shows:

$$g^{\mathrm{mess}} = \frac{1}{\mathrm{H}} \sum_{h=1}^{\mathrm{H}} g_h^{\mathrm{embed}} \hat{\mathcal{A}}_h. \tag{28}$$

On this basis, the updated feature is given as follows:

$$g^{\mathrm{update}} = xW^{\mathrm{id}} + b^{\mathrm{id}} + g^{\mathrm{mess}}. \tag{29}$$

Finally, the label is predicted through a multi-layer perceptron (MLP) with a length of L in the following manner:

$$\hat{y} = \left\{\left[...\sigma(g^{\mathrm{update}}W^1 + b^1)\right] W^{\mathrm{L}} + b^{\mathrm{L}}\right\}, \tag{30}$$

where the superscript indicates the layer number, and σ is the activated function and set as ReLU function in this study:

$$\sigma(x) \triangleq \max\{0, x\}. \tag{31}$$

Based on the model inference results, Fig. 3 (b) proposes the $\mathcal{A}_h|_{h=1}^{\mathrm{H}}$ learning procedure during model training stage. For model training stage, the mean-squared-error is selected as the learning objective function:

$$\mathcal{L}(y,\mathcal{A},x) \triangleq \frac{1}{\mathrm{N}_{\mathcal{B}}} \sum_{l=1}^{\mathrm{N}_{\mathcal{B}}} \|\hat{y}_l - y_l\|^2, \tag{32}$$

where $\mathcal{B}$ denotes batch size. Based on this, the learning objective $\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)$ is realized by the automatic differential-based DL backends. Notably, for simplicity, the kernel function is selected as the radial basis function defined as follows:

$$\mathcal{K}(\mathcal{A},\mathcal{A}') = \exp(-\frac{1}{h}|\mathcal{A}-\mathcal{A}'|^2), \tag{33}$$

where the bandwidth h is set as the median value of $\mathcal{A}|_{h=1}^{\mathrm{H}}$ based on previous reference [40]. In order to complete the article, the detailed training and testing algorithms are given in Section S.I.E of supplementary material.

E. Theoretical Analysis of KEMS Algorithm

In this subsection, the convergence, uniqueness, and stability of the KEMS algorithm for E^{2}AG learning are discussed in detail to uphold the rigor of this manuscript.

As mentioned in Section IV-A, matrix $\mathcal{A}$ can be decomposed into row vector $\vec{a}_i|_{i=1}^{\mathrm{M}}$. To facilitate reading, the row vector $\vec{a}_i$, where $i \in \{1,2,...,\mathrm{M}\}$ is adopted to analyze the convergence of algorithm in this subsection. Before proposing the analysis, it is necessary to clarify concepts. First, the row vector $\vec{a}_i|_{i=1}^{\mathrm{M}}$ belongs to the standard simplex $\Delta^{\mathrm{M}-1}$. On this basis, the projected operator $\nabla\psi(\cdot)$ based on function $\psi(\vec{a}_i) = \sum_{j=1}^{\mathrm{M}} \vec{a}_{i,j}\log\vec{a}_{i,j} - a_{i,j}$ can be defined as $\nabla\psi(\vec{a}_i) = \log\vec{a}_i$.

On this basis, the convergence of $\mathcal{L}_{\mathrm{ER}}$ is obtained by proving the learning procedure defined in (26) can result in a *monotonic decreasing yet lower bounded* $\mathcal{L}_{\mathrm{ER}}$. Thus, the following proposition is given:

Proposition 3. *Let the KEMS algorithm be defined as per the iteration process in (26). The convergence of the KEMS algorithm to a stable state is guaranteed under the condition that the learning rate, denoted by η, is sufficiently small.*

Proof. The proof is given in Section S.I.D of supplementary material. □

After that, the following proposition about the algorithm's uniqueness is given based on Proposition 3:

Proposition 4. *Assume the functional $\mathcal{L}_{ER}$ exhibits λ-convexity. Given two distinct initial values, $\vec{a}_i^0$ and $\vec{\alpha}_i^0$, the evolution trajectories of these values at time τ adhere to the following inequality:*

$$|\vec{a}_i^\tau - \vec{\alpha}_i^\tau| \leq |\vec{a}_i^0 - \vec{\alpha}_i^0| \times \exp(-\lambda\tau). \tag{34}$$

Proof. The proof is given in Section S.I.D of supplementary material. □

Based on this, suppose $\vec{\alpha}_i^0$ is at the equilibrium point, where the condition $\frac{\mathrm{d}\mathcal{L}_{\mathrm{ER}}}{\mathrm{d}\vec{\alpha}_i^\tau} = 0$ holds, the following corollary about the stability around equilibrium point can be given:

Corollary 5. *Suppose $\vec{\alpha}_i^0$ is positioned at an equilibrium point of the functional $\mathcal{L}_{ER}$, characterized by the condition $\frac{\mathrm{d}\mathcal{L}_{ER}}{\mathrm{d}\vec{\alpha}_i^\tau} = 0$. Under these circumstances, the following exponential stability property is established:*

$$|\vec{a}_i^\tau - \vec{\alpha}_i^0| \leq |\vec{a}_i^0 - \vec{\alpha}_i^0| \times \exp(-\lambda\tau).$$

Proof. The proof is given in Section S.I.D of supplementary material. □

Up to now, the convergence analysis of the KEMS algorithm for E^{2}AG learning is established theoretically.

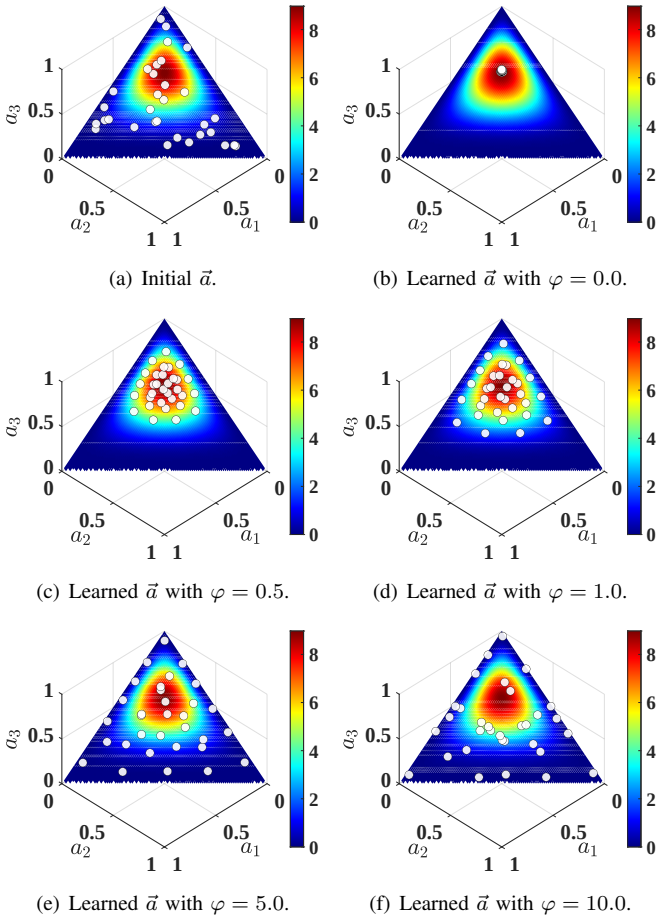
V. Experimental Results & Discussions

In this section, the following questions are answered to validate the effectiveness of the proposed approach empirically:

- **Influence:** *What's the influence of the entropy regularization term on the learned graphs?* Section V-A presents the performance of the learned graphs with the existence of the entropy regularization (ER) term empirically to validate the necessity of the ER term.
- **Accuracy:** *Does E^{2}AG work?* Section V-C evaluates E^{2}AG's performance against a variety of baseline methods using four datasets from real industrial scenarios, thereby establishing a foundational understanding of its operational efficacy.
- **Complexity:** *How efficient is the E^{2}AG model?* Section V-D proposes the training and testing time of E^{2}AG compared to the selected baseline models using four datasets from real industrial scenarios, thereby establishing a demonstration of its operational efficiency.
- **Convergence:** *Does the KEMS algorithm for E^{2}AG converge?* To back up the proofs in Section IV-E, Section V-E proposes the model performance along training epochs, thereby empirically proving the convergence of the KEMS algorithm.
- **Gains:** *Why does E^{2}AG work?* Section V-F deconstructs E^{2}AG to discern the sources of its performance gain.
- **Sensitivity:** *Is it sensitive to key hyperparameters?* Section V-G elucidates the impact of different hyperparameters on the prediction accuracy, analyzing the E^{2}AG's responsiveness to parameter alterations.

All experiments are conducted on a workstation equipped with an Intel Xeon E5 processor, 8 Nvidia GTX 1080 GPUs, and 128 GB of RAM. To maintain consistency and fairness across evaluations, the Adam optimizer, as detailed by [43], is employed uniformly across all experimental runs. The model training and inference processes are carried out using Python 3.8 and PyTorch 1.13 [17], respectively. Each experiment is replicated a minimum of three times, employing four different random seeds to ensure the robustness and reproducibility of the results.

A. Influence of ER Term



(a) Initial $\vec{a}$. (b) Learned $\vec{a}$ with $\varphi = 0.0$.

(c) Learned $\vec{a}$ with $\varphi = 0.5$. (d) Learned $\vec{a}$ with $\varphi = 1.0$.

(e) Learned $\vec{a}$ with $\varphi = 5.0$. (f) Learned $\vec{a}$ with $\varphi = 10.0$.

Fig. 4. Influence of ER term experiment, where the contour indicates the logarithmic pdf of $\mathrm{Dir}([2.5, 2.5, 5.0])$, and the white scatters indicate the learned $\vec{a}$ varying different ER strength φ.

This subsection answers the question 'What's the influence of the ER term on the learned graphs?' by presenting an empirical validation of the KEMS algorithm, demonstrating its capability to enhance the diversification of the learned graph structure. Notably, the regression error term $\mathcal{L}(y, \mathrm{H}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x), x)$ is modeled following a Gaussian distribution as per (32). According to Bayesian principles, $\mathcal{L}(y, \mathrm{H}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x), x) \propto \log p(\mathcal{A}|x, y)p(\mathcal{A})$, where the expression on the right-hand side represents an unnormalized negative log-probability density function (pdf) associated with a Dirichlet distribution. Through this framework, the impact of the ER term is elucidated by comparing the learned graphs $\mathcal{A}|_{h=1}^{\mathrm{H}}$ with the pdf of a predefined Dirichlet distribution.

To this end, consider one row $\vec{a}_h|_{h=1}^{\mathrm{H}}$ from graph $\mathcal{A}_h|_{h=1}^{\mathrm{H}}$. *Assume* the cost functional is defined as the negative log-

likelihood of a three-dimensional Dirichlet distribution:

$$\mathcal{L}(y, \mathrm{H}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h, x), x) \propto -\sum_{h=1}^{\mathrm{H}}\left\{\log\frac{\Gamma\left(\sum_{k=1}^{3}\rho_k\right)}{\prod_{k=1}^{3}\Gamma(\rho_k)}+\sum_{k=1}^{3}(\rho_k-1)\log a_{k,h}\right\},$$

where the concentration parameter $[\rho_1, \rho_2, \rho_3]$ is defined as $[2.5, 2.5, 5.0]$, and $\Gamma(\cdot)$ denotes the gamma function. On this foundation, setting the learning rate η to 5.0×10^{-3}, the head number H as 32, and the iteration count to 100, the optimized result of the learned graph $\vec{a}$ is illustrated in Fig. 4. From the contour in Fig. 4, it is evident that the $\vec{a}$ makes the functional with high-density value form a region rather than a precise point. Based on these observations, as the strength of the ER term, denoted as φ, increases, the learned graph initially collapses to a single point (as shown in Fig. 4 (b)). It then gradually aligns along the contour (as depicted in Fig. 4 (c) and (d)), and ultimately tends to distribute uniformly across the entire simplex (as illustrated in Fig. 4 (e) and (f)). This progression indicates that the ER term effectively regularizes the learned graph towards a more uniform distribution, thereby naturally enhancing its diversification. However, excessively high values of φ can lead the learned graph to deviate from the true optimal values, potentially impairing the model's performance.

B. Experimental Protocol

1) Industrial Process Description: The Catalysis Shift Conversion (CSC) unit, Carbon-Dioxide Absabsorption Column (CAC), Methanation Furance Unit (MFU), and Primary Reformer Unit (PRU) from a real ammonia synthesis process are selected to demonstrate the effectiveness of E^{2}AG model. The flowsheets and other detailed information are provided in Section S.II of the supplementary material.

2) Baseline Model Selection: To provide a thorough comparison and showcase the effectiveness of E^{2}AG model, the baseline models categorized as follows are chosen for prediction accuracy comparison:

- **MHSAM-based Models:** Flashformer [44], iTransformer [45], Random Synthesizer (Synthesizer(Random)) [46], and Dense Synthesizer (Synthesizer(Dense)) [46].
- **AGNN-based Models:** Deep Learning model with Dynamic Graph (DGDL) [16] and stacked graph convolutional network (S-GCN) [15]
- **Fix Graph Models:** Two-Stream Graph Convolutional Network-Incorporated Latent Feature Analysis (TGLFA) [24], Symmetry-Preserving Dual-Stream Graph Neural Networks (SDGNN) [47], Adaptive Graph Contrastive Learning (AdaGCL) [21], Graph Convolution Network with Random Weight (GCN-RW) [48], and UniFilter [49].

The reasons for choosing these baseline models and other detailed information about hyperparameters for these models are given in Section S.III of supplementary material.

3) Evaluation Metrics: To evaluate the model performance, the RMSE, R^2, MAE, and MAPE in (35) to (38) are adopted, where the N_{test} and $\bar{y}$ are the size of the testing dataset and average value of label y, respectively. For RMSE, MAE, and MAPE, the closer value is to 0, the more accurate the prediction. While for R^2, the closer R^2 is to 1, the better the prediction performance of model.

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}_{\text{test}}}\sum_{l=1}^{\mathrm{N}_{\text{test}}}(y_l-\hat{y}_l)^2} \tag{35}$$

$$\mathrm{R}^2 = 1-\frac{\sum_{l=1}^{\mathrm{N}_{\text{test}}}(y_l-\hat{y}_l)^2}{\sum_{l=1}^{\mathrm{N}_{\text{test}}}(y_l-\bar{y})^2} \tag{36}$$

$$\mathrm{MAPE} = \frac{1}{\mathrm{N}_{\text{test}}}\sum_{l=1}^{\mathrm{N}_{\text{test}}}\left|\frac{(y_l-\hat{y}_l)}{y_l}\right|\times100\% \tag{37}$$

$$\mathrm{MAE} = \frac{1}{\mathrm{N}_{\text{test}}}\sum_{l=1}^{\mathrm{N}_{\text{test}}}|y_l-\hat{y}_l| \tag{38}$$

C. Baseline Comparison

In this subsection, the question 'Does E^{2}AG work?' is investigated. To this end, the baseline comparison results are given in Table I. Based on Table I, the following observations can be summarized:

1) MHSAM-based models and AGNN-based models generally exhibit more stable performance compared to fixed-graph models across various scenarios.
2) Almost all models perform better on the CSC dataset than on the CAC, MFU, and PRU datasets on R^2 metric.
3) For the CSC dataset, the E^{2}AG model surpasses the baseline models across several metrics, improving R^2 by 0.33% to 40529%, reducing RMSE by 2.48% to 64.18%, lowering MAPE by 3.42% to 96.82%, and decreasing MAE by 3.48% to 64.2%.
4) For the CAC dataset, the E^{2}AG model surpasses most of the baseline models across several metrics, improving R^2 by 0.57% to 35091%, reducing RMSE by 0.84% to 51.38%, lowering MAPE by 0.75% to 46.03%, and decreasing MAE by 0.75% to 46.57%.
5) For the MFU dataset, the E^{2}AG model surpasses most of the baseline models across several metrics, improving R^2 by 1.08% to 1290.84%, reducing RMSE by 3.55% to 47.69%, lowering MAPE by 2.74% to 73.22%, and decreasing MAE by 1.36% to 50.28%.
6) For the PRU dataset, the E^{2}AG model surpasses most of the baseline models across several metrics, improving R^2 by 0.53% to 17972.9%, reducing RMSE by 0.41% to 33.33%, lowering MAPE by 1.87% to 36.29%, and decreasing MAE by 1.5% to 34.53%.

Observation 1) underscores that adaptively treating the graph structure in GNNs can significantly enhance model performance, justifying the application of AGNNs in industrial soft sensor tasks. After that, Observation 2) indicates that the performance of the soft sensor heavily depends on the datasets.

TABLE I
BASELINE COMPARISON RESULTS

Model / Dataset	CSC				CAC			
Metric	R^2	RMSE	MAPE	MAE	R^2	RMSE	MAPE	MAE
Flashformer	6.644E-1	2.699E-1	1.103E-1	2.167E-1	7.597E-1	3.569E-3	9.651E-1	2.808E-3
iTransformer	9.395E-1	1.452E-1	5.851E-2	1.151E-1	7.494E-1†	3.646E-3†	9.797E-1	2.854E-3
Synthesizer (Random)	9.306E-1	1.549E-1	6.194E-2	1.218E-1	7.427E-1†	3.696E-3†	9.976E-1†	2.905E-3†
Synthesizer (Dense)	9.339E-1	1.516E-1	6.110E-2†	1.201E-1†	7.507E-1	3.636E-3	9.855E-1	2.869E-3
S-GCN	9.332E-1†	1.527E-1†	6.184E-2†	1.216E-1†	6.553E-1†	4.278E-3†	1.163E0†	3.387E-3†
TGLFA	9.377E-1†	1.475E-1†	5.893E-2†	1.159E-1†	6.756E-1†	4.148E-3†	1.110E0†	3.231E-3†
SDGNN	8.993E-1†	1.871E-1†	7.317E-2†	1.438E-1†	7.441E-1†	3.684E-3†	**9.423E-1**	**2.750E-3**
AdaGCL	5.490E-1†	3.953E-1†	1.579E-1†	3.103E-1†	2.081E-1†	6.479E-3†	1.646E0†	4.827E-3†
GCN-RW	2.320E-3†	**7.278E-3**†	1.776E0†	**5.219E-3**†	2.171E-3†	7.279E-3†	1.775E0†	5.216E-3†
UniFilter	8.318E-1†	2.413E-1†	9.625E-2†	1.891E-1†	5.354E-1†	4.966E-3†	1.321E0†	3.853E-3†
DGDL	9.341E-1	1.516E-1	6.099E-2	1.199E-1	7.288E-1†	3.794E-3†	1.023E0†	2.980E-3†
E^2AG	**9.426E-1**	1.416E-1	**5.651E-2**	1.111E-1	**7.640E-1**	**3.539E-3**	9.579E-1	2.787E-3
Win Counts	11	10	11	10	11	11	10	10

Model / Dataset	MFU				PRU			
Metric	R^2	RMSE	MAPE	MAE	R^2	RMSE	MAPE	MAE
Flashformer	7.066E-1†	2.167E-1†	8.130E+1	1.684E-1†	5.593E-1†	9.591E-1†	1.890E+1†	7.377E-1†
iTransformer	7.158E-1†	2.132E-1†	7.625E+1	1.663E-1†	4.648E-1†	1.057E0†	2.030E+1†	8.042E-1†
Synthesizer (Random)	6.886E-1†	2.232E-1†	7.687E+1†	1.748E-1†	3.370E-1†	1.176E0†	2.285E+1†	8.977E-1†
Synthesizer (Dense)	7.137E-1†	2.140E-1†	7.566E+1†	1.675E-1†	4.425E-1†	1.079E0†	2.091E+1†	8.271E-1†
S-GCN	6.860E-1†	2.241E-1†	8.448E+1†	1.757E-1†	3.292E-1†	1.184E0†	2.306E+1†	9.014E-1†
TGLFA	7.121E-1†	2.146E-1†	**6.720E+1**	1.662E-1†	4.791E-1†	1.042E0†	2.057E+1†	7.982E-1†
SDGNN	7.215E-1†	2.111E-1†	6.738E+1†	1.638E-1†	5.880E-1†	9.279E-1†	1.802E+1†	7.171E-1†
AdaGCL	4.596E-1†	2.939E-1†	1.815E+2†	2.358E-1†	1.104E-1†	1.363E0†	2.720E+1†	1.060E0†
GCN-RW	5.327E-2†	3.892E-1†	2.748E+2†	3.204E-1†	3.347E-3†	**7.275E-3**†	**1.774E0**†	**5.213E-3**†
UniFilter	6.535E-1†	2.354E-1†	1.097E+2†	1.854E-1†	2.705E-1†	1.235E0†	2.467E+1†	9.562E-1†
DGDL	7.330E-1†	2.067E-1†	7.711E+1	1.615E-1	6.017E-1	9.124E-1	1.766E+1	7.046E-1
E^2AG	**7.409E-1**	**2.036E-1**	7.359E+1	**1.593E-1**	**6.049E-1**	9.087E-1	1.733E+1	6.940E-1
Win Counts	11	11	9	11	11	10	10	10

† marks the variants that E^2AG outperforms significantly at p-value < 0.05 over paired samples t-test. **Bolded** results indicate the best in each metric. Wavy results indicate the second best in each metric.

This can be understood from the perspective of industrial operation scenarios. For example, during the ammonia synthesis process, the input gas of the CAC may be contaminated with diverse impurities like hydrogen sulfide stemming from natural gas feedstock. Consequently, the outlet concentration of carbon monoxide may be significantly affected, introducing substantial noise into the dataset. As a result, the R^2 score on the CSC dataset is better than that on the CAC dataset. Finally, Observations 3) to 6) demonstrate that, despite being an AGNN-based model, the E^2AG model outperforms most baselines. This underscores the potential of enhancing AGNN-based models with ensemble mechanism and entropy regularization mechanisms.

D. Computational Complexity

Following the detailed assessment of model performance, this subsection delves into the analysis of each model's training and testing durations. Table II outlines the theoretical time complexity for a range of baseline models. From the table, it is evident that models such as E^2AG, which incorporate ensemble mechanisms similar to MHSAM-based models like the iTransformer, Flashformer, and both Synthesizer variants, exhibit a time complexity that may increase with the number of heads. However, unlike traditional MHSAM-based models that typically utilize dot-product similarity, E^2AG does not formulate its graph using this method. As a result, the time complexity of E^2AG aligns with those of the Synthesizer (Dense) and Synthesizer (Random) models, and is half that of the iTransformer. This suggests a more efficient computational performance under specific configurations. Moreover, it is noteworthy that other GNNs such as S-GCN, TGLFA, SDGNN, and DGDL, exhibit a time complexity that scales

TABLE II
TIME COMPLEXITY COMPARISON RESULTS

Model Name	Time Complexity
Flashformer	$\mathcal{O}\{\mathrm{H}[\frac{\mathrm{D}^2 d^{\text{embed}}}{\text{Block Size}} + \mathrm{D}(\text{Block Size})d^{\text{embed}}]\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
iTransformer	$\mathcal{O}\{2\mathrm{HD}^2 d^{\text{embed}}\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
Synthesizer (Random)	$\mathcal{O}\{\mathrm{HD}^2 d^{\text{embed}}\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
Synthesizer (Dense)	$\mathcal{O}\{\mathrm{HD}^2 d^{\text{embed}}\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
S-GCN	$\mathcal{O}\{\mathrm{HD}^2 d^{\text{embed}}\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
TGLFA	$\mathcal{O}\{\mathrm{N_{hop}D}(d^{\text{embed}})^2\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
SDGNN	$\mathcal{O}\{\mathrm{N_{hop}D}(d^{\text{embed}})^2\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
AdaGCL	$\mathcal{O}\{\mathrm{N_{hop}D}(d^{\text{embed}})^2\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
GCN-RW	$\mathcal{O}\{2\mathrm{D}^2 d^{\text{embed}} + 2(d^{\text{embed}})^2\mathrm{D}\mathcal{B}\}$
UniFilter	$\mathcal{O}\{2\mathrm{N_{hop}D}\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
DGDL	$\mathcal{O}\{\mathrm{N_{hop}}[\mathrm{HD}^2 d^{\text{embed}}\mathcal{B}]\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$
E^2AG	$\mathcal{O}\{\mathrm{HD}^2 d^{\text{embed}}\mathcal{B}\} + \mathcal{O}[\mathcal{B}(\sum_{l=1}^{\mathrm{L}-1} h^l h^{l+1})]$

For industrial soft sensor modeling, the graph formed by process variables is considerably smaller compared to scenarios like recommender systems. Therefore, the analysis assumes that the adjacency matrix $\mathcal{A}$ is dense, and $\mathrm{N_{hop}}$ represents the number of hops.

TABLE III
COMPUTATIONAL TIME COMPARISON RESULTS VARY DIFFERENT MODELS

Model Name	CSC		CAC		MFU		PRU	
	Train Time	Test Time	Train Time	Test Time	Train Time	Test Time	Train Time	Test Time
Flashformer	1.025E+2	5.440E-2	9.433E+1	4.340E-2	4.607E+1	2.386E-2	5.012E+1	2.836E-2
iTransformer	8.127E+1	3.712E-2	7.618E+1	3.308E-2	3.949E+1	2.038E-2	4.029E+1	1.713E-2
Synthesizer (Random)	5.858E+1	3.698E-2	5.327E+1	3.223E-2	3.020E+1	1.541E-2	2.744E+1	1.470E-2
Synthesizer (Dense)	5.449E+1	2.982E-2	4.680E+1	2.536E-2	2.522E+1	1.380E-2	2.541E+1	1.377E-2
S-GCN	2.538E+2	5.394E-2	2.146E+2	5.054E-2	1.167E+2	2.432E-2	1.123E+2	2.398E-2
TGLFA	7.828E+1	3.992E-2	6.352E+1	3.213E-2	2.987E+1	1.590E-2	3.657E+1	1.838E-2
SDGNN	1.524E+2	6.578E-2	1.129E+2	5.382E-2	5.668E+1	2.530E-2	6.121E+1	2.626E-2
AdaGCL	1.305E+2	4.568E-2	1.086E+2	3.576E-2	5.264E+1	1.890E-2	5.825E+1	2.084E-2
GCN-RW	**1.133E-2**	**1.559E-2**	**1.099E-2**	**1.274E-2**	**1.166E-2**	**7.040E-3**	**1.090E-2**	**6.716E-3**
UniFilter	7.470E+1	3.623E-2	6.246E+1	2.940E-2	3.292E+1	1.514E-2	3.530E+1	1.693E-2
DGDL	1.455E+2	7.056E-2	1.346E+2	5.934E-2	6.584E+1	2.901E-2	6.592E+1	3.119E-2
E^2AG	7.635E+1	2.560E-2	6.681E+1	2.461E-2	3.298E+1	1.144E-2	3.901E+1	1.255E-2

The unit is second. **Bolded** and wavy results indicate the smallest and the second smallest result.

with the square of the embedding dimension and the number of hops. This scaling indicates that if the dimensionality D of the data is less than the square of the embedding dimension d^{embed}, models like Flashformer, iTransformer, and the Synthesizers, as well as E^2AG, could potentially offer faster computation compared to the aforementioned GNNs. Conversely, for higher-dimensional data, these GNNs may be less computationally efficient.

Based on the abovementioned analysis, the computational times for various models on the experimental datasets are detailed in Table III. From the table, it is clear that GNN-RW exhibits the shortest training and testing times across different datasets. This efficiency is primarily due to GNN-RW lacking a downstream MLP model, thereby eliminating the need for gradient-descent-based training and inference complexity brought by MLP. Among MHSAM-based models such as Flashformer, iTransformer, and Synthesizers, it is evident that their computational times are generally lower than those of the traditional GNNs. This observation supports the earlier analysis regarding time complexity, highlighting the efficiency of transformer architectures in handling such tasks. The training time for SDGNN is noted to be longer than those of most baseline models, attributed to its requisite pre-training stage prior to the actual S-GCN training. This pre-training phase introduces additional computational overhead, impacting overall training duration. Lastly, the training time for E^2AG falls in the mid-range, while its testing time is the second shortest among the models evaluated. This performance positions E^2AG as a viable option for industrial soft sensor applications, balancing computational efficiency with effective model operation.

E. Empirical Investigation of Convergence

This subsection explores whether the KEMS algorithm converges empirically and answers the question 'Does the KEMS algorithm for E^2AG converge?'. Fig. 5 depicts the model's mean square error (MSE) on the training dataset as the number of epochs increases. Based on this, the following observations can be given:

1) As the number of training epochs increases, the MSE for all models shows a consistent and monotonic decrease. This trend underscores the algorithm's ability to effectively minimize the training loss over time, thereby confirming its practical efficacy in model learning.
2) Concurrent with the increase in epochs, the shaded area representing the variability of MSE across different runs diminishes. Eventually, this variability stabilizes, indicating that the models reach a plateau. This behavior empirically validates the convergence property of the algorithm, as delineated in Proposition 3.
3) Notably, the models tend to stabilize within approximately 50 epochs. This rapid convergence suggests that despite variations in initial conditions, the models reliably settle to a stable state, which further validates the uniqueness and stability of the KEMS algorithm, as delineated in Proposition 4 and Corollary 5.

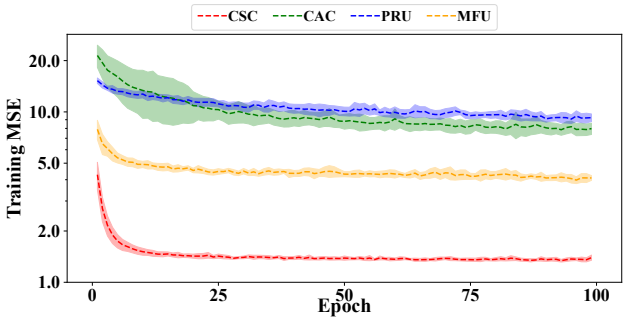


Fig. 5. The mse curve along epoch. The shaded area indicates the ± 0.75 standard deviation uncertainty interval.

These observations collectively affirm that the KEMS algorithm for E^2AG learning not only reduces MSE effectively but also exhibits strong convergence characteristics, confirming both its effectiveness and stability in practical applications.

F. Ablation Study

Building on the insights gained from Section V-C, the focus of this subsection is to delve deeper into the factors contributing to the superior performance of the E^2AG model, specifically addressing 'Why does E^2AG work?'. To explore this question, an ablation study was conducted, which emphasized three critical components: the ensemble mechanism (by

TABLE IV
ABLATION STUDY RESULTS

Ablate Module			CSC				CA			
MD	EM	ER	$R^2(\downarrow)$	RMSE(↑)	MAE(↑)	MAPE(↑)	$R^2(\downarrow)$	RMSE(↑)	MAE(↑)	MAPE(↑)
✓	✗	✗	0.4002%	3.1382%	3.7956%	3.8193%	1.7736%	2.7085%	2.0258%	2.1109%
✓	✗	✓	0.6317%	4.8031%	5.2993%	5.3356%	2.0442%	3.0919%	2.2813%	2.2750%
✓	✓	✗	0.0219%	0.1766%	0.7987%	0.8039%	1.3706%	2.0759%	1.4022%	1.4339%
✗	✗	✗	0.6752%	4.8899%	5.5647%	5.5928%	1.3229%	2.0332%	1.0118%	1.0046%
✗	✗	✓	0.4382%	3.3788%	4.0457%	4.0706%	3.0305%	4.3912%	3.7591%	3.8110%
✗	✓	✗	0.2485%	1.9654%	2.8924%	2.9172%	1.5287%	2.3597%	1.6969%	1.6747%
✗	✓	✓	0.5748%	4.4117%	5.0163%	5.0418%	1.8664%	2.8266%	1.7389%	1.6964%
Ablate Module			MFU				PRU			
MD	EM	ER	$R^2(\downarrow)$	RMSE(↑)	MAE(↑)	MAPE(↑)	$R^2(\downarrow)$	RMSE(↑)	MAE(↑)	MAPE(↑)
✓	✗	✗	2.0647%	2.7729%	5.4176%	2.1487%	3.5025%	2.4880%	3.1190%	1.3319%
✓	✗	✓	2.6027%	3.4181%	2.4732%	3.2291%	3.5867%	2.5235%	4.5539%	2.2728%
✓	✓	✗	1.3226%	1.8042%	1.0775%	1.6179%	3.1772%	2.2195%	4.0819%	1.6968%
✗	✗	✗	1.7120%	2.3075%	2.6406%	1.6042%	3.5361%	2.4769%	2.1846%	1.2242%
✗	✗	✓	4.3190%	5.2008%	3.2969%	4.7669%	5.0510%	3.4478%	4.7493%	2.7987%
✗	✓	✗	2.5781%	3.3888%	6.8545%	2.9409%	8.6321%	5.5384%	6.4862%	5.5315%
✗	✓	✓	2.8505%	3.7070%	2.3028%	2.9087%	9.4007%	5.7784%	7.3980%	5.3817%

setting the number of heads, H, to 1, abbreviated as 'EM'), the entropy regularization term (replacing (26) with (12), abbreviated as 'ER'), and the mirror descent-based graph learning strategy (substituting (26) with a standard gradient descent-based optimizer, Adam [43], abbreviated as 'MD'). The results of this study are presented in Table IV.

Initially, considering scenarios where the MD is not ablated, data from Table IV clearly indicates that omitting both the ensemble mechanism and the entropy regularization term significantly diminishes model performance. This finding highlights the integral role of both components in boosting the model's efficacy. Notably, removing only the ensemble mechanism appears to have a more pronounced negative impact on performance than removing both components. This can be interpreted from a gradient descent perspective: when the number of heads is reduced to one, kernel regularization adversely affects model training through a biased gradient descent, leading to performance degradation. Conversely, when only the entropy regularization term is omitted, performance decline is evident but less severe compared to the other scenarios. This underlines the ensemble mechanism's pivotal role in augmenting the E^2AG model's performance. Moreover, this also showcases the beneficial impact of incorporating the entropy regularization term during the ensemble process.

Furthermore, when considering the scenario where MD is ablated, it is evident from Table IV that the absence of the MD term consistently results in more pronounced degradation than scenarios where MD is retained. This phenomenon highlights the critical importance of the row normalization constraint during the AGNN model training phase and further validates the use of the mirror descent-based learning strategy for AGNN training. In summary, the outcomes of the ablation study strongly support the significance of integrating the ensemble mechanism, entropy regularization term, and mirror-descent-based strategy into the E^2AG training protocol, demonstrating their collective importance in ensuring optimal model performance.

G. Sensitivity Analysis

In this subsection, the question 'Is the model sensitive to key hyperparameters?' is investigated. The hyperparameters in focus are the learning rate η, batch size $\mathcal{B}$, and head number H. Adjustments to these hyperparameters were made, and the corresponding results are presented in Figs. 6 to 9. Based on the findings from Figs. 6 to 9, the following observations can be made:

1) As the learning rate increases, the model performance significantly declines for all four datasets.
2) As the batch size increases, the model performance decreases on the CSC, PRU, and MFU datasets, while improving on the CAC dataset.
3) As the head number increases, the model performance tends to initially improve and then decline.
4) As the strength of the ER term increases, model performance initially improves but subsequently declines.

Observation 1) suggests that the algorithm is sensitive to the learning rate, with a smaller learning rate potentially leading to better model performance. Conversely, Observation 2) indicates that the optimal batch size is highly dependent on the dataset properties. As mentioned earlier, the CAC process may suffer from noise issues due to impurities in the feedstock, and thus a larger batch size may help the model mitigate the impact of dataset noise. However, for the CSC, PRU, and MFU datasets, which may not suffer from noise issues, a larger batch size could lead to overgeneralization and subsequent performance degradation. After that, Observation 3) implies that the head number should be greater than 1 to enhance model performance. However, an excessive number of heads may lead to overfitting and reduced model performance. In conclusion, these findings demonstrate that it is crucial to employ a smaller learning rate during training, suitable batch size, and head number to ensure optimal performance. Finally, Observation 4) highlights the impact of the ER term. Initially, the inclusion of the ER term can enhance model

performance, aligning with the motivation of this manuscript. However, as the strength of the ER term increases, the learned model parameters may deviate from the optimal, as seen in the numerical experiments detailed in Section V-A, leading to a decline in performance. In conclusion, these findings underscore the importance of carefully selecting a smaller learning rate, an appropriate batch size, an optimal number of heads, and a suitable ER strength to ensure peak model performance. The observations also validate the theoretical motivation behind this manuscript, emphasizing the nuanced balance required in parameter tuning.

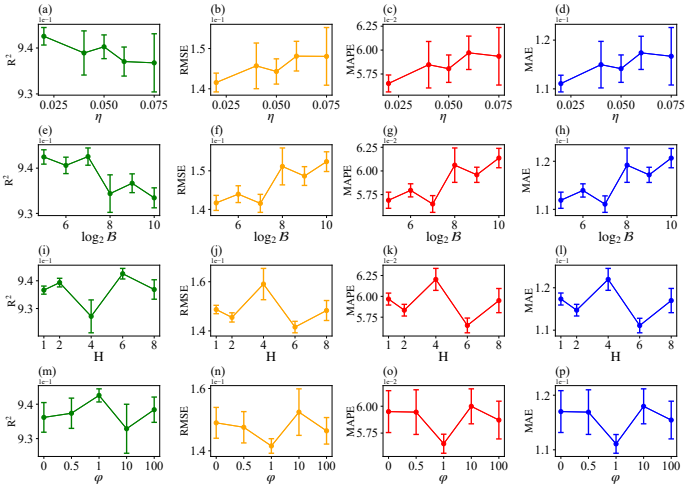


Fig. 6. The soft sensor accuracy given varying values of learning rate η (a-d), batch size $\mathcal{B}$ (e-h), head number H (i-l), and ER strength φ (m-p) on CSC dataset. The panels indicate R^2, RMSE, MAPE and MAE from left to right. The error bar indicates the $\pm$ 0.5 standard deviation uncertainty interval.

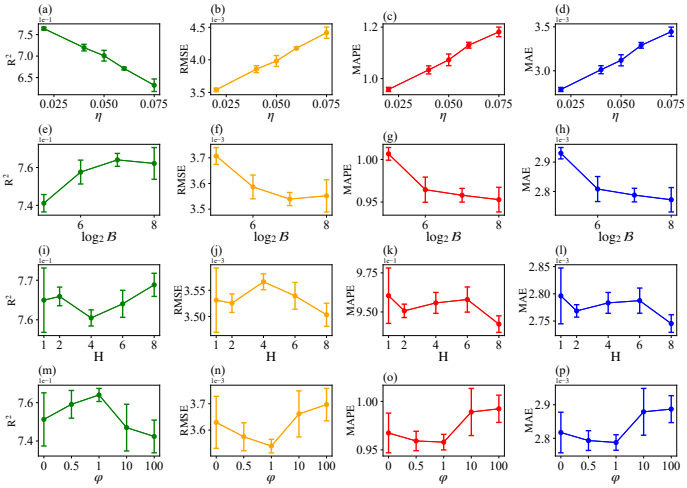


Fig. 7. The soft sensor accuracy given varying values of learning rate η (a-d), batch size $\mathcal{B}$ (e-h), head number H (i-l), and ER strength φ (m-p) on CAC dataset. The panels indicate R^2, RMSE, MAPE and MAE from left to right. The error bar indicates the $\pm$ 0.5 standard deviation uncertainty interval.

VI. Conclusions

In this study, to investigate the feasibility of learning row-normalized graphs within the gradient descent-based DL backends and improve the AGNN model efficiency, a novel model named E^{2}AG was proposed. Initially, the learning of row-normalized graph was recast into a mirror descent problem, and the discrepancy metric was replaced with the entropy function term to promise the model learning process

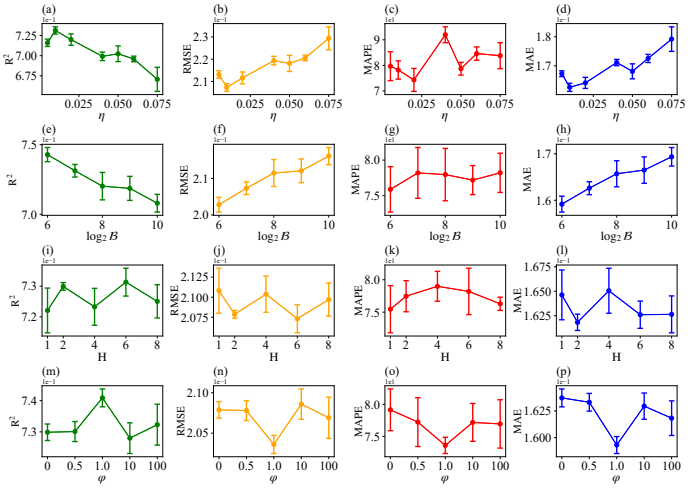


Fig. 8. The soft sensor accuracy given varying values of learning rate η (a-d), batch size $\mathcal{B}$ (e-h), head number H (i-l), and ER strength φ (m-p) on MFU dataset. The panels indicate R^2, RMSE, MAPE and MAE from left to right. The error bar indicates the $\pm$ 0.5 standard deviation uncertainty interval.

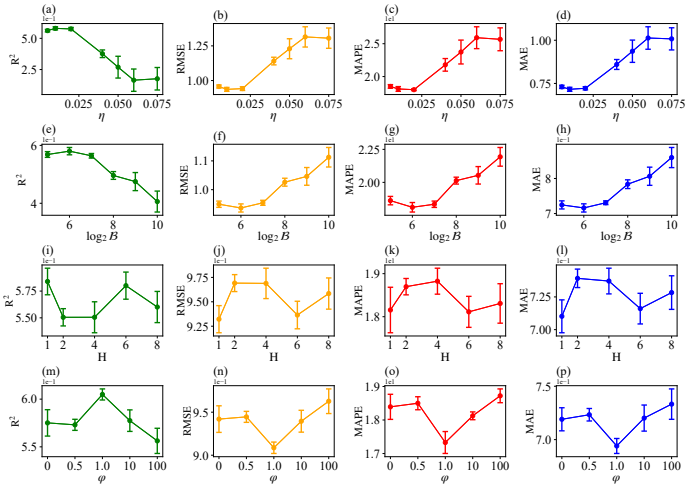


Fig. 9. The soft sensor accuracy given varying values of learning rate η (a-d), batch size $\mathcal{B}$ (e-h), head number H (i-l), and ER strength φ (m-p) on PRU dataset. The panels indicate R^2, RMSE, MAPE and MAE from left to right. The error bar indicates the $\pm$ 0.5 standard deviation uncertainty interval.

satisfies the row normalization constraint. After that, based on the analysis of MHSAM, the ensemble AGNN structure was proposed, and the entropy regularization term was then added to improve the ensemble efficiency. On this basis, for implementation simplicity, the closed-form expressions for E^{2}AG learning were further derived. Consequently, the architecture and corresponding training & testing algorithms of E^{2}AG were summarized. Finally, the efficacy of the E^{2}AG model was validated through its application to two inferential sensor tasks.

There are, however, several issues that remain unresolved. Firstly, the function class assumption of the velocity field presents a limitation. In this study, for model implementation, the functional class is restricted within the RKHS, which may lead to an incorrect functional gradient direction when the initial locations of the graphs are significantly distant from the optimal graphs [50]. Secondly, the iterative procedure assumes a continuity equation within the Wasserstein space [51], when the data exhibit 'unbalanced' multi-modal characteristics (some modal has high density and some modal has low density) [22, 52], such as in cases where an industrial process has

newly-established steady state data, the proposed KEMS algorithm for E^2AG learning may lose its effectiveness. Finally, exploring the use of Riemannian gradient descent [53, 54] to develop a simpler and faster graph learning procedure, or converting the proposed KEMS algorithm into a plug-in-and-play framework and extending it to other larger deep learning structures/larger network structure present significant research opportunities.

ACKNOWLEDGMENTS

Zhichao Chen, the first author, would like to extend his sincere gratitude to Mr. Fangyikang Wang of Zhejiang University and Mr. Wenjian Du of Sun Yat-sen University for their valuable discussions on particle variational inference and convergence analysis. He is also grateful to Dr. Chang Liu from Microsoft for inspiring related works on mirror descent and Riemannian gradient descent approaches.

REFERENCES

[1] F. Qian, Y. Jin, S. J. Qin, and K. Sundmacher, "Guest editorial special issue on deep integration of artificial intelligence and data science for process manufacturing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3294–3295, 2021.

[2] C. Liu, Y. Wang, C. Yang, and W. Gui, "Multimodal data-driven reinforcement learning for operational decision-making in industrial processes," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 1, pp. 252–254, 2024.

[3] J. Qian, L. Jiang, and Z. Song, "Locally linear back-propagation based contribution for nonlinear process fault diagnosis," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 764–775, 2020.

[4] X. Jiang, X. Kong, and Z. Ge, "Augmented industrial data-driven modeling under the curse of dimensionality," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 6, pp. 1445–1461, 2023.

[5] H. Xu, Z. Liu, H. Wang, C. Li, Y. Niu, W. Wang, and X. Liu, "Denoising diffusion straightforward models for energy conversion monitoring data imputation," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2024.

[6] Q. Sun and Z. Ge, "Gated stacked target-related autoencoder: A novel deep feature extraction and layerwise ensemble method for industrial soft sensor application," *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3457–3468, 2022.

[7] K. Wang, C. Shang, L. Liu, Y. Jiang, D. Huang, and F. Yang, "Dynamic Soft Sensor Development Based on Convolutional Neural Networks," *Industrial & Engineering Chemistry Research*, vol. 58, no. 26, pp. 11 521–11 531, Jul. 2019.

[8] C. Li, G. Li, Y. Song, Q. He, Z. Tian, H. Xu, and X. Liu, "Fast forest fire detection and segmentation application for uav-assisted mobile edge computing system," *IEEE Internet of Things Journal*, vol. 11, no. 16, pp. 26 690–26 699, 2024.

[9] L. Yao and Z. Ge, "Dynamic features incorporated locally weighted deep learning model for soft sensor development," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[10] C. Zhang, J. Yella, Y. Huang, X. Qian, S. Petrov, A. Rzhetsky, and S. Bom, "Soft sensing transformer: Hundreds of sensors are worth a single word," in *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 1999–2008.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems(NeuralIPS 2017)*, 2017, p. 6000–6010.

[12] M. Li, A. Micheli, Y. G. Wang, S. Pan, P. Lió, G. S. Gnecco, and M. Sanguineti, "Guest editorial: Deep neural networks for graphs: Theory, models, algorithms, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 4367–4372, 2024.

[13] M. Jia, Y. Dai, D. Xu, T. Yang, Y. Yao, and Y. Liu, "Deep graph network for process soft sensor development," in *2021 8th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, 2021, pp. 1–6.

[14] Y. Wang, P. Yan, and M. Gai, "Dynamic soft sensor for anaerobic digestion of kitchen waste based on sgstgat," *IEEE Sensors Journal*, vol. 21, no. 17, pp. 19 198–19 208, 2021.

[15] Y. Wang, Q. Sui, C. Liu, K. Wang, X. Yuan, and G. Dong, "Interpretable prediction modeling for froth flotation via stacked graph convolutional network," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 1, pp. 334–345, 2024.

[16] K. Zhu and C. Zhao, "Dynamic graph-based adaptive learning for online industrial soft sensor with mutable spatial coupling relations," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 9, pp. 9614–9622, 2023.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 1–12.

[18] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne *et al.*, "JAX: Autograd and XLA," *Astrophysics Source Code Library*, pp. ascl–2111, 2021.

[19] W. Liu, X. Zheng, J. Su, L. Zheng, C. Chen, and M. Hu, "Contrastive proxy kernel stein path alignment for cross-domain cold-start recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11 216–11 230, 2023.

[20] M. Jing, Y. Zhu, T. Zang, J. Yu, and F. Tang, "Graph contrastive learning with adaptive augmentation for recommendation," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2022, pp. 590–605.

[21] Y. Jiang, C. Huang, and L. Huang, "Adaptive graph contrastive learning for recommendation," in *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 2023, pp. 4252–4261.

[22] H. Wang, J. Fan, Z. Chen, H. Li, W. Liu, T. Liu, Q. Dai, Y. Wang, Z. Dong, and R. Tang, "Optimal transport for treatment effect estimation," in *Advances in Neural Information Processing Systems (NeuralIPS' 23)*, vol. 36. Curran Associates, Inc., 2023, pp. 5404–5418.

[23] H. Li, Y. Xiao, C. Zheng, P. Wu, Z. Geng, X. Chen, and P. Cui, "Debiased collaborative filtering with kernel-based causal balancing," in *International Conference on Learning Representations*, 2024, pp. 1–12.

[24] F. Bi, T. He, Y. Xie, and X. Luo, "Two-stream graph convolutional network-incorporated latent feature analysis," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 3027–3042, 2023.

[25] M. Jia, D. Xu, T. Yang, Y. Liu, and Y. Yao, "Graph convolutional network soft sensor for process quality prediction," *Journal of Process Control*, vol. 123, pp. 12–25, 2023.

[26] R. Zhai, J. Zeng, and Z. Ge, "Structured principal component analysis model with variable correlation constraint," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 2, pp. 558–569, 2022.

[27] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 1907–1913.

[28] B. N. Oreshkin, A. Amini, L. Coyle, and M. Coates, "FC-GAGA: Fully connected gated graph architecture for spatio-temporal traffic forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 10, 2021, pp. 9233–9241.

[29] Y. Huang, C. Zhang, J. Yella, S. Petrov, X. Qian, Y. Tang, X. Zhu, and S. Bom, "Grassnet: Graph soft sensing neural networks," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 746–756.

[30] H. Wang, Z. Wang, Y. Niu, Z. Liu, H. Li, Y. Liao, Y. Huang, and X. Liu, "An accurate and interpretable framework for trustworthy process monitoring," *IEEE Transactions on Artificial Intelligence*, pp. 1–12, 2023.

[31] Z. Liu, Y. Cao, H. Xu, Y. Huang, Q. He, X. Chen, X. Tang, and X. Liu, "Hidformer: Hierarchical dual-tower transformer using multi-scale mergence for long-term time series forecasting," *Expert Systems with Applications*, vol. 239, p. 122412, 2024.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR 2016)*, 2016, Conference Proceedings, pp. 1–14.

[33] X. Hong, T. Zhang, Z. Cui, and J. Yang, "Variational gridded graph convolution network for node classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1697–1708, 2021.

[34] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30(NIPS'17)*, 2017, Conference Proceedings, pp. 1025–1035.

[35] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR 2018)*, 2018, Conference Proceedings, pp. 1–12.
[36] Y. Ma and J. Tang, *Deep Learning on Graphs*. Cambridge University Press, 2020.
[37] J. Darzentas, "Problem complexity and method efficiency in optimization," *Journal of the Operational Research Society*, vol. 35, no. 5, pp. 455–455, 1984.
[38] J. Shi, C. Liu, and L. Mackey, "Sampling with mirrored Stein operators," *International Conference on Learning Representations*, pp. 1–26, 2022.
[39] S. Patterson and Y. W. Teh, "Stochastic gradient Riemannian langevin dynamics on the probability simplex," *Advances in neural information processing systems*, vol. 26, pp. 1–9, 2013.
[40] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," *Advances in neural information processing systems*, vol. 29, 2016.
[41] Y. Li and R. E. Turner, "Gradient estimators for implicit models," in *International Conference on Learning Representations*, 2018, pp. 1–13.
[42] D. Wang and Q. Liu, "Nonlinear stein variational gradient descent for learning diversified mixture models," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6576–6585.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015, pp. 1–8.
[44] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems (NeurIPS 2022)*, 2022, pp. 1–16.
[45] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "iTransformer: Inverted transformers are effective for time series forecasting," in *The Twelfth International Conference on Learning Representations*, 2024, pp. 1–22.
[46] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng, "Synthesizer: Rethinking self-attention for transformer models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 183–10 192.
[47] J. Chen, Y. Yuan, and X. Luo, "SDGNN: Symmetry-preserving dual-stream graph neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 7, pp. 1717–1719, 2024.
[48] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu, "Are graph convolutional networks with random weights feasible?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 2751–2768, 2023.
[49] K. Huang, Y. G. Wang, M. Li, and P. Lio, "How universal polynomial bases enhance spectral graph neural networks: Heterophily, over-smoothing, and over-squashing," in *Proceedings of the 41st International Conference on Machine Learning (ICML' 24)*, ser. Proceedings of Machine Learning Research, vol. 235. PMLR, 21–27 Jul 2024, pp. 20 310–20 330.
[50] H. Dong, X. Wang, L. Yong, and T. Zhang, "Particle-based variational inference with preconditioned functional gradient flow," in *The Eleventh International Conference on Learning Representations*, 2022, pp. 1–26.
[51] F. Wang, H. Zhu, C. Zhang, H. Zhao, and H. Qian, "GAD-PVI: A general accelerated dynamic-weight particle-based variational inference framework," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 1–29.
[52] H. Wang, T.-W. Chang, T. Liu, J. Huang, Z. Chen, C. Yu, R. Li, and W. Chu, "ESCM^2: Entire space counterfactual multi-task model for post-click conversion rate estimation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 363–372. [Online]. Available: https://doi.org/10.1145/3477495.3531972
[53] C. Liu and J. Zhu, "Riemannian stein variational gradient descent for bayesian inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[54] R. Zhang, Q. Liu, and X. Tong, "Sampling in constrained domains with orthogonal-space variational gradient descent," *Advances in Neural Information Processing Systems (NeurIPS 2022)*, vol. 35, pp. 37 108–37 120, 2022.

Zhichao Chen received the B.Eng. degree in chemical engineering and technology from School of Chemical Engineering and Technology, Sun Yat-sen University, Zhuhai, China, in 2020. He is currently working toward the Ph.D. degree in control science and engineering with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, China.

His research interests include process data analytics, both linear and nonlinear optimization, Bayesian inference, differential equation, and variational methods.

Licheng Pan received the B.Eng. in Automation from the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2021 and is currently pursuing his Master's degree in Control Science and Engineering at the College of Control Science and Engineering, Zhejiang University, Hangzhou, China.

His research interests include multi-task learning, multivariate soft sensor modeling, and time series analysis.

Yiran Ma received the B.Eng. degree in automation from the School of Automation Science and Engineering, South China University of Technology, Guangzhou, China, in 2021. He is currently working toward the Ph.D. degree in control science and engineering with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, China.

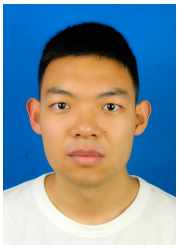
His current research interests include machine learning, Bayesian methods, and their applications in industrial data-driven modeling.

Zeyu Yang (M'23) received the Ph.D. degree from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2021. From Sep. 2019 to Mar. 2020, he was a Visiting Scholar with the Department of Automatic Control and Complex Systems (AKS), University of Duisburg-Essen, Duisburg, Germany.

He is currently an Associate Professor with the School of Engineering, Huzhou University, Huzhou, China. His research interests include industrial big data, process monitoring, soft sensor, data-driven modeling, process data analysis, and their industrial applications.

Le Yao (M'20) received the B. Eng. and M. Eng. degrees in Automation from the Department of Control Science and Engineering, Jiangnan University, Wuxi, China, in 2012 and 2015, respectively, and received the Ph.D. degree in Automation from the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2019. He was a Post-Doctoral Research Fellow with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, from May 2019 to Mar 2022. From July 2023 to September 2023, he was a visiting scholar with the Hong Kong University of Science and Technology, Hong Kong. He is currently an Associate Professor with the School of Mathematics, Hangzhou Normal University, Hangzhou.

His research interests include industrial big data, process monitoring, soft sensor, data-driven modeling, distributed computing, process data analysis, and their industrial applications.



Jinchuan Qian (M'24) received the B.Eng. degree in automation from Hefei University of Technology, Anhui, China, in 2017, and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2022.

He is currently a Postdoctoral Research Fellow with the State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou, China. His research interests include fault detection, fault diagnosis, and industrial big data modeling.

Zhihuan Song received the B. Eng. and M. Eng. degrees in industrial automation from the Hefei University of Technology, Anhui, China, in 1983 and 1986, respectively, and the Ph. D. degree in industrial automation from Zhejiang University, Hangzhou, China, in 1997. Since 1997, he has been with the Department of Control Science and Engineering, Zhejiang University, where he was first a Postdoctoral Research Fellow, then an Associate Professor, and is currently a Professor. He has published more than 200 papers in journals and conference proceedings.

His research interests include the modeling and fault diagnosis of industrial processes, analytics and applications of industrial big data, and advanced process control technologies.

Supplementary Material for E^{2}AG: Entropy-Regularized Ensemble Adaptive Graph for Industrial Soft Sensor Modeling

S.I. Derivation of Concerning Equations

A. Perturbation Over Loss Function

Based on the celebrated Taylor's expansion, the perturbation over loss function $\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h + \eta\phi(\mathcal{A}_h), x), x)$ can be expanded as follows:

$$\begin{aligned}
&\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h + \eta\phi(\mathcal{A}_h), x), x)\\
=&\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h), x)\\
&+ \underbrace{\eta\frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} [\nabla_{\mathcal{A}_h}\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h), x)]^\top \phi(\mathcal{A}_h)}_{\text{term 2}}\\
&+ O(\eta^2).
\end{aligned} \tag{S1}$$

Note that, the term 2 can be further reformulated as:

$$\begin{aligned}
&\eta\frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} [\nabla_{\mathcal{A}_h}\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h), x)]^\top \phi(\mathcal{A}_h)\\
=&\eta\mathbb{E}_{\rho(\mathcal{A})}\{[\nabla_{\mathcal{A}}\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h), x)]^\top \phi(\mathcal{A})\}
\end{aligned} \tag{S2}$$

B. Perturbation Over Entropy Function

In this subsection, concerning derivative for loss function computation is given. For simplicity, the perturbation of $\mathcal{A}$ is denoted as follows:

$$\mathcal{T}(\mathcal{A}) = \mathcal{A} + \eta\phi(\mathcal{A}), \tag{S3}$$

where η and and $\phi(\mathcal{A})$ is called infinitesimal parameter and variation. When η is small enough, $\mathcal{T}$ is a one-to-one map and $|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))| \neq 0$ holds. Consequently, based on the celebrated change of variables formula [SR1, SR2], the transformed density function $\rho_{\mathcal{T}}(\mathcal{A})$ can be derived by the original density function $\rho(\mathcal{A})$ as follows:

$$\rho_{\mathcal{T}}(\mathcal{A})|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))| = \rho(\mathcal{A}). \tag{S4}$$

And thus, the following equations for entropy function is obtained:

$$\begin{aligned}
&\mathbb{H}[\rho_{\mathcal{T}}(\mathcal{A})] - \mathbb{H}[\rho(\mathcal{A})]\\
=&-\int \rho_{\mathcal{T}}(\mathcal{A})\log\rho_{\mathcal{T}}(\mathcal{A})\mathrm{d}\mathcal{A} + \int \rho(\mathcal{A})\log\rho(\mathcal{A})\mathrm{d}\mathcal{A}\\
=&-\int \rho_{\mathcal{T}}(\mathcal{T}(\mathcal{A}))\log\rho_{\mathcal{T}}(\mathcal{T}(\mathcal{A}))\mathrm{d}\mathcal{T}(\mathcal{A})\\
&+\int \rho(\mathcal{A})\log\rho(\mathcal{A})\mathrm{d}\mathcal{A}\\
\overset{\text{(i)}}{=}&-\int \rho(\mathcal{A})\log\frac{\rho(\mathcal{A})}{|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|}\mathrm{d}\mathcal{A}\\
&+\int \rho(\mathcal{A})\log\rho(\mathcal{A})\mathrm{d}\mathcal{A}\\
=&\int \rho(\mathcal{A})\log\left(|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|\right)\mathrm{d}\mathcal{A}\\
=&\mathbb{E}_{\rho(\mathcal{A})}[\log\left(|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|\right)],
\end{aligned} \tag{S5}$$

where (i) is based on (S4):

$$\begin{aligned}
&\int \rho_{\mathcal{T}}(\mathcal{T}(\mathcal{A}))\log\rho_{\mathcal{T}}(\mathcal{T}(\mathcal{A}))\mathrm{d}\mathcal{T}(\mathcal{A})\\
=&\int \frac{\rho(\mathcal{A})}{|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|}\\
&\times\log\frac{\rho(\mathcal{A})}{|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|}|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|\mathrm{d}\mathcal{A}
\end{aligned} \tag{S6}$$

C. Functional Derivative

Based on (S2), the following equation can be obtained:

$$\begin{aligned}
&\frac{\mathrm{d}\eta\mathbb{E}_{\rho(\mathcal{A})}\{[\nabla_{\mathcal{A}}\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h), x)]^\top\phi(\mathcal{A})\}}{\mathrm{d}\eta}|_{\eta=0}\\
=&\mathbb{E}_{\rho(\mathcal{A})}\{[\nabla_{\mathcal{A}}\mathcal{L}(y, \frac{1}{\mathrm{H}}\sum_{h=1}^{\mathrm{H}} f_\theta(\mathcal{A}_h), x)]^\top\phi(\mathcal{A})\}.
\end{aligned} \tag{S7}$$

Analogously, based on the last line of (S5), the following equation can be obtained:

$$\begin{aligned}
&\frac{\mathrm{d}\mathbb{E}_{\rho(\mathcal{A})}[\log\left(|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|\right)]}{\mathrm{d}\eta}|_{\eta=0}\\
=&\mathbb{E}_{\rho(\mathcal{A})}[\frac{1}{|\det(\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A}))|}\frac{\mathrm{d}\nabla_{\mathcal{A}}\mathcal{T}(\mathcal{A})}{\mathrm{d}\eta}]|_{\eta=0}\\
=&\mathbb{E}_{\rho(\mathcal{A})}[1\times\nabla_{\mathcal{A}}\phi(\mathcal{A})]|_{\eta=0}\\
=&\mathbb{E}_{\rho(\mathcal{A})}[\nabla_{\mathcal{A}}\phi(\mathcal{A})].
\end{aligned} \tag{S8}$$

And thus, the following equation is obtained:

$$\begin{aligned}&\frac{\mathrm{d}[\mathcal{L}_{\mathrm{ER}}(y,\mathcal{A}+\eta\phi(\mathcal{A}),x)-\mathcal{L}_{\mathrm{ER}}(y,\mathcal{A},x)]}{\mathrm{d}\eta}|_{\eta=0}\\ =&\mathbb{E}_{\rho(\mathcal{A})}[\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^{\top}\phi(\mathcal{A})-\nabla_{\mathcal{A}}\phi(\mathcal{A})].\end{aligned} \tag{S9}$$

D. Concerning Proofs

Proposition (2)**.** *Assume that the perturbation direction $\phi(\mathcal{A})$ is restricted in RKHS with kernel function $\mathcal{K}(x,y)$, (24) can be reformulated as follows:*

$$\mathbb{E}_{\rho(\mathcal{A})}[-\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^{\top}\mathcal{K}(\mathcal{A},\cdot)+\nabla_{\mathcal{A}}\mathcal{K}(\mathcal{A},\cdot)]. \tag{S10}$$

Proof. Assume there exists a feature map $\xi(x):\mathbb{R}^{\mathrm{d}}\to\mathcal{H}$, the kernel function $\mathcal{K}(x,y)$ is defined as $\mathcal{K}(x,y)=\langle\xi(x),\xi(y)\rangle_{\mathcal{H}}$. Denote the regularization function $\mathcal{Q}(\phi)$ as $\mathcal{Q}(\phi)=\frac{1}{2}\|\phi\|_{\mathcal{H}}$. The spectral decomposition of kernel function can be given by $\mathcal{K}(x,y)=\sum_{i=1}^{\infty}\zeta_i\xi_i(x)\xi_i(y)$, where $\xi_i:\mathbb{R}^{\mathrm{d}}\to\mathbb{R}$ are orthonormal basis and ζ_i is the corresponding eigenvalue. On this basis, the function within this space $\phi\in\mathcal{H}^{\mathrm{d}}$ can be decomposed as: $\phi(x)=\sum_{i=1}^{\infty}\phi_i\sqrt{\zeta_i}\xi_i(x)$, where $\phi_i\in\mathbb{R}^{\mathrm{d}}$ and $\sum_{i=1}^{\infty}\|\phi_i\|^2<\infty$ according to references [SR3–SR6]. The optimization objective function based on the right-hand-side of (24) can be defined as follows based on the celebrated steepest descent method:

$$\begin{aligned}\hat{\phi}=&\arg\min_{\phi\in\mathcal{H}^{\mathrm{d}}}\int\rho(\mathcal{A})[\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^{\top}\phi(\mathcal{A})-\nabla_{\mathcal{A}}\phi(\mathcal{A})]\mathrm{d}\mathcal{A}\\&+\mathcal{Q}(\phi)\\=&\arg\min_{\phi\in\mathcal{H}^{\mathrm{d}}}\int\rho(\mathcal{A})[\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^{\top}\phi(\mathcal{A})-\nabla_{\mathcal{A}}\phi(\mathcal{A})]\mathrm{d}\mathcal{A}\\&+\frac{1}{2}\|\phi\|^2_{\mathcal{H}^{\mathrm{d}}}\\=&\arg\min_{\phi\in\mathcal{H}^{\mathrm{d}}}\int\rho(\mathcal{A})[\sum_{i=1}^{\infty}\sqrt{\zeta_i}\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^{\top}\phi_i\xi_i(\mathcal{A})\\&-\nabla_{\mathcal{A}}\sum_{i=1}^{\infty}\phi_i\sqrt{\zeta_i}\xi_i(\mathcal{A})]\mathrm{d}\mathcal{A}+\frac{1}{2}\sum_{i=1}^{\infty}\|\phi_i\|^2\end{aligned} \tag{S11}$$

Take the partial derivative of the last row with respect to ϕ_i and set it equal to 0, $\hat{\phi}_i$ can be given as follows:

$$\hat{\phi}_i=\sqrt{\zeta_i}\int\rho(\mathcal{A})[\nabla_{\mathcal{A}}\xi_i(\mathcal{A})-\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)\xi_i(\mathcal{A})]\mathrm{d}\mathcal{A}. \tag{S12}$$

Finally, $\hat{\phi}(x)$ can be obtained as follows:

$$\begin{aligned}\hat{\phi}(\mathcal{A})=&\sum_{i=1}^{\infty}\hat{\phi}_i\sqrt{\zeta_i}\xi_i(\mathcal{A})\\=&\int\rho(\mathcal{A})[-\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^{\top}\mathcal{K}(\mathcal{A},\cdot)+\nabla_{\mathcal{A}}\mathcal{K}(\mathcal{A},\cdot)]\mathrm{d}\mathcal{A}.\end{aligned} \tag{S13}$$

□

Proposition (3)**.** *Let the KEMS algorithm be defined as per the iteration process in (26). The convergence of the KEMS algorithm to a stable state is guaranteed under the condition that the learning rate, denoted by η, is sufficiently small.*

Proof. The proof is categorized into three steps as follows:

1) **Relationship between $\Delta^{\mathrm{M}-1}$ and $\mathbb{R}^{\mathrm{M}}$**: The following expression for the mirror descent defined in (6) can be obtained:

$$\begin{aligned}&w^{t+1}=\arg\min_{w\in\Omega}\{(\nabla_w\mathcal{L}|_{w=w^t})^{\top}w+\frac{1}{\eta}\mathbb{B}\left[w\|w^t\right]\}\Rightarrow\\&0=\nabla_w\mathcal{L}|_{w=w^t}+\nabla_w\frac{\mathbb{B}[w\|w^t]}{\eta}|_{w=w^{t+1}}\Rightarrow\\&\frac{\mathrm{d}\nabla\psi(w)}{\mathrm{d}\tau}=\lim_{\eta\to\infty}\frac{\nabla\psi(w^{t+1})-\nabla\psi(w^t)}{\eta}=-\nabla_w\mathcal{L}.\end{aligned} \tag{S14}$$

From the last line, it can be observed that modeling the mirror descent can be treated as a discretization of ordinary differential equation (ODE) with Euler's method [SR7] with step size η. Notably, for simplex constraint, $\nabla\psi(\vec{a}_i)=\log\vec{a}_i$. Consequently, the following equality holds:

$$\begin{aligned}&\frac{\mathrm{d}\vec{a}_i}{\mathrm{d}\tau}\times\mathrm{MakeDiag}(\vec{a}_i)^{-1}=\frac{\mathrm{d}\log\vec{a}_i}{\mathrm{d}\tau}\\&=-\mathbb{E}_{\rho(\vec{a}_i)}[\nabla_{\vec{a}_i}\mathcal{L}(y,\vec{a}_i,x)^{\top}\mathcal{K}(\vec{a}_i,\cdot)-\nabla_{\vec{a}_i}\mathcal{K}(\vec{a}_i,\cdot)],\end{aligned} \tag{S15}$$

where MakeDiag converts an M-dimensional row vector into a size-M × M diagonal matrix.

2) **Decreasing Conidition:** Assume we want to decrease the $\mathcal{L}_{\mathrm{ER}}$ along τ, we can have the following expression to delineate the $\mathcal{L}_{\mathrm{ER}}$ along τ:

$$\begin{aligned}&\frac{\mathrm{d}\mathcal{L}_{\mathrm{ER}}}{\mathrm{d}\tau}=\frac{\mathrm{d}\mathcal{L}_{\mathrm{ER}}}{\mathrm{d}\vec{a}_i}[\frac{\mathrm{d}\vec{a}_i}{\mathrm{d}\tau}]^{\top}=\frac{\mathrm{d}\mathcal{L}_{\mathrm{ER}}}{\mathrm{d}\vec{a}_i}\mathrm{MakeDiag}(\vec{a}_i)[\frac{\mathrm{d}\log\vec{a}_i}{\mathrm{d}\tau}]^{\top}\\&=-\|\mathbb{E}_{\rho(\vec{a}_i)}[\nabla_{\vec{a}_i}\mathcal{L}(y,\vec{a}_i,x)^{\top}\mathcal{K}(\vec{a}_i,\cdot)-\nabla_{\vec{a}_i}\mathcal{K}(\vec{a}_i,\cdot)]\|_2^2\\&\times\prod_{j=1}^{\mathrm{M}}\vec{a}_{i,j}\leq 0.\end{aligned} \tag{S16}$$

Based on this, it can be seen that the evolution of $\vec{a}_i$ progressively decreases the functional $\mathcal{L}_{\mathrm{ER}}$.

3) **Bounded Condition:** In this part, the bounded condition of $\mathcal{L}_{\mathrm{ER}}$ is checked. At the beginning, the following inequality is given:

$$\mathcal{L}_{\mathrm{ER}}(y,\mathcal{A},x)\geq 0-\mathrm{M}\sum_{j=1}^{\mathrm{M}-1}\log(j)=-\mathrm{M}\sum_{j=1}^{\mathrm{M}-1}\log(j), \tag{S17}$$

where the inequality is based on the following inequalities:

$$\frac{1}{\mathrm{N}_{\mathcal{B}}}\sum_{l=1}^{\mathrm{N}_{\mathcal{B}}}\|\hat{y}_l-y_l\|^2\geq 0,$$

and

$$\begin{aligned}&\mathbb{D}_{\mathrm{KL}}[\rho(\vec{a}_i)\|\mathcal{U}(\vec{a}_i)]\\&=-\mathbb{H}(\rho(\vec{a}_i))-\mathbb{E}_{\rho(\vec{a}_i)}[\log\mathcal{U}(\vec{a}_i)]\geq 0\\&\Rightarrow-\mathbb{H}(\rho(\vec{a}_i))\geq\mathbb{E}_{\rho(\vec{a}_i)}[\log\mathcal{U}(\vec{a}_i)]=\log\mathcal{U}(\vec{a}_i),\end{aligned}$$

and $\mathcal{U}(\vec{a}_i)$ is the uniform on $\Delta^{\mathrm{M}-1}$, whose density is a constant $\mathcal{U}(\vec{a}_i)=\prod_{j=1}^{\mathrm{M}-1}j^{-1}$.

From steps 2) and 3), it can be concluded that the iteration given in (26) is monotonic decreasing and lower bounded by a constant. Based on this fact, the proposed iterative expression for E^{2}AG is converged. Building upon this foundation, a lower

learning rate η, which corresponds to the step size in Euler's method, results in the iteration curve of $\mathcal{L}_{\text{ER}}$ more closely approximating the ODE defined in (S16). Consequently, a lower learning rate η yields a sequence of the learning objective $\mathcal{L}_{\text{ER}}$ that monotonically decreases. □

Proposition (4). *Assume the functional $\mathcal{L}_{ER}$ exhibits λ-convexity. Given two distinct initial values, $\vec{a}_i^0$ and $\vec{\alpha}_i^0$, the evolution trajectories of these values at time τ adhere to the following inequality:*

$$|\vec{a}_i^\tau - \vec{\alpha}_i^\tau| \leq |\vec{a}_i^0 - \vec{\alpha}_i^0| \times \exp(-\lambda\tau). \tag{S18}$$

Proof. Recall (S15), the following equation can be obtained:

$$\frac{\mathrm{d}\vec{\gamma}_\tau}{\mathrm{d}\tau} = -\left(\frac{\mathrm{d}\mathcal{L}_{\text{ER}}}{\mathrm{d}\vec{a}_i^\tau} - \frac{\mathrm{d}\mathcal{L}_{\text{ER}}}{\mathrm{d}\vec{\alpha}_i^\tau}\right), \tag{S19}$$

where $\vec{\gamma}_\tau$ is defined as follows:

$$\vec{\gamma}_\tau = \log \vec{a}_i^\tau - \log \vec{\alpha}_i^\tau. \tag{S20}$$

Since $\mathcal{L}_{\text{ER}}$ is λ-convex, the following inequality is obtained:

$$\frac{\mathrm{d}\mathcal{L}_{\text{ER}}}{\mathrm{d}\vec{a}_i^\tau} - \frac{\mathrm{d}\mathcal{L}_{\text{ER}}}{\mathrm{d}\vec{\alpha}_i^\tau} \geq \lambda|\vec{a}_i^\tau - \vec{\alpha}_i^\tau|, \tag{S21}$$

hence the following inequality can be obtained:

$$\frac{\mathrm{d}\vec{\gamma}_\tau}{\mathrm{d}\tau} \leq -\lambda|\vec{a}_i^\tau - \vec{\alpha}_i^\tau|. \tag{S22}$$

According to the celebrated Lagrange mean value theorem, for $\vec{a}_i^\tau, \vec{\alpha}_i^\tau \in \Delta^{\text{M}-1}$, the following inequality holds:

$$1 \leq \frac{|\log \vec{a}_i^\tau - \log \vec{\alpha}_i^\tau|}{|\vec{a}_i^\tau - \vec{\alpha}_i^\tau|} \Rightarrow |\vec{a}_i^\tau - \vec{\alpha}_i^\tau| \leq |\log \vec{a}_i^\tau - \log \vec{\alpha}_i^\tau|. \tag{S23}$$

Consequently, the following inequality is obtained by applying (S23) to the left-hand-side of (S22):

$$\frac{\mathrm{d}|\vec{a}_i^\tau - \vec{\alpha}_i^\tau|}{\mathrm{d}\tau} \leq -\lambda|\vec{a}_i^\tau - \vec{\alpha}_i^\tau|. \tag{S24}$$

Therefore, (34) is proved according to the celebrated Gronwall's lemma [SR6]. □

Corollary (5). *Suppose $\vec{\alpha}_i^0$ is positioned at an equilibrium point of the functional $\mathcal{L}_{ER}$, characterized by the condition $\frac{\mathrm{d}\mathcal{L}_{ER}}{\mathrm{d}\vec{\alpha}_i^\tau} = 0$. Under these circumstances, the following exponential stability property is established:*

$$|\vec{a}_i^\tau - \vec{\alpha}_i^0| \leq |\vec{a}_i^0 - \vec{\alpha}_i^0| \times \exp(-\lambda\tau).$$

Proof. Note that, the condition $\frac{\mathrm{d}\mathcal{L}_{\text{ER}}}{\mathrm{d}\vec{\alpha}_i^\tau} = 0$ holds for stability point. Consequently, the following equation can be obtained:

$$\vec{\alpha}_i^\tau = \vec{\alpha}^0. \tag{S25}$$

On this basis, plug (S25) into (S18) the corollary is proved. □

E. Algorithm Summary

Algorithm S1 and S2 summarize the model inference and corresponding training & testing algorithm for E^{2}AG, respectively. From Algorithm S2, it can be seen that the key for E^{2}AG is modifying the iterative procedure for graph $\mathcal{A}_h|_{h=1}^{\text{H}}$ during model training.

Algorithm S1: The E^{2}AG inference Algorithm

Input: Process Variables: $\{x_1, x_2, ..., x_n\}$
Parameter: Model Parameter set θ ($\mathcal{A} \subset \theta$)
Output: Qualtiy Variable:$\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$

`/* Feature Preprocessing */`
$g_{i,h}^{\text{embed}} \leftarrow x_i \times W_{i,h}^{\text{embed}} + b_{i,h}^{\text{embed}}|_{i=1}^{\text{N}_{\text{node}}}$;
`/* Message Exchange & Aggregation */`
$g^{\text{mess}} \leftarrow \frac{1}{\text{H}} \sum_{h=1}^{\text{H}} g_h^{\text{embed}} \hat{\mathcal{A}}_h$;
`/* Node Information Update */`
$g^{\text{update}} \leftarrow xW^{\text{id}} + b^{\text{id}} + g^{\text{mess}}$;
Label Prediction Part:
$\hat{y} \leftarrow \left\{\left[...\sigma(g^{\text{update}}W^1 + b^1)\right]W^{\text{L}} + b^{\text{L}}\right\}$

S.II. Dataset Descriptions

In this section, the background of the Catalysis Shift Conversion (CSC) unit, Carbon-Dioxide Absabsorption Column (CAC), Methanation Furance Unit (MFU), and Primary Reformer Unit (PRU) are delineated. The rest of this section will focus on introducing concerning background of these two industrial processes.

A. Carbon-Dioxide Absabsorption Column

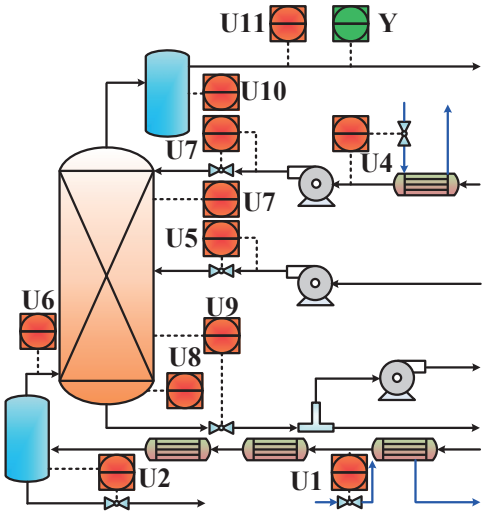


Fig. S1. The flowsheet of CAC

Fig. S1 presents the flowsheet of the CAC [SR8]. The carbon-dioxide absorber is a vital piece of equipment in ammonia synthesis process to handle the carbon-dioxide by-product in the hydrogen from upstream unit. The sodium hydroxide solvent is chosen to be absorption liquid and the corresponding chemical reaction can be given in (S26):

$$\begin{cases} \mathrm{CO_2 + 2KOH = K_2CO_3 + H_2O} \\ \mathrm{K_2CO_3 + H_2O + CO_2 = 2KHCO_3} \end{cases}. \tag{S26}$$

To eliminate the carbon-dioxide concentration in the hydrogen stream for promoting the product quality in downstream urea synthesis process, the carbon-dioxide concentration in the outlet gas should be monitored in real time. However,

Algorithm S2: Training & Testing Algorithm of E^{2}AG

Input: Train, validate, and test data: $\{x_m, y_m\}_{m\in\mathcal{D}_{\text{train}}}$, $\{x_v, y_v\}_{v\in\mathcal{D}_{\text{valid}}}$, $\{x_n, y_n\}_{n\in\mathcal{D}_{\text{test}}}$

Hyperparameters:

Batch size: $\mathcal{B}$, learning Rate: η, Epoch: $\mathcal{E}$, parameter θ (note that $\log\vec{a}\in\theta$), head number H.

Training:

Set epoch $= 1$

while *epoch* $\leq \mathcal{E}$ **do**

/* Forward Computation */

Sample a minibatch $\mathcal{D}_{\text{minibatch}} \subset \mathcal{D}_{\text{train}}$;

$\{\hat{y}_i\}_{i=1}^{\mathcal{B}} \leftarrow$ Algorithm S1;

/* Computation of $\log p(y|\hat{\mathcal{A}}, x)$ */

$$\mathcal{L}(y,\mathcal{A},x) \leftarrow \frac{1}{\mathrm{N}_{\mathcal{B}}}\sum_{l=1}^{\mathrm{N}_{\mathcal{B}}}\|\hat{y}_l - y_l\|^2$$

/* Gradient with respect to θ */

Obtain gradient: $\nabla_\theta\mathcal{L}$

/* Modify Gradient $\log\mathcal{A}$ */

$$e^{-\lambda_i} \leftarrow \sum_{j=1}^{\mathrm{M}} \mathcal{A}_{i,j}\exp\{-\eta[\mathbb{E}_{\rho(\mathcal{A})}[\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^\top \times \mathcal{K}(\mathcal{A},\cdot) - \nabla_{\mathcal{A}}\mathcal{K}(\mathcal{A},\cdot)]|_{\mathcal{A}=\mathcal{A}^t}]_{i,j}\}$$

$$\log\vec{a}_i^{t+1} = \log\vec{a}_i^t + \lambda_i - \eta \times \{\mathbb{E}_{\rho(\mathcal{A})}[\begin{matrix}\nabla_{\mathcal{A}}\mathcal{L}(y,\mathcal{A},x)^\top\mathcal{K}(\mathcal{A},\cdot)\\ -\nabla_{\mathcal{A}}\mathcal{K}(\mathcal{A},\cdot)\end{matrix}]|_{\mathcal{A}=\mathcal{A}^t}\}_i$$

$$\nabla_{\log\vec{a}_i}\log p(\hat{\mathcal{A}}|\mathcal{D}) \leftarrow \frac{1}{\vec{a}_i}\nabla_{\log\vec{a}_i}\log p(\hat{\mathcal{A}}|\mathcal{D})$$

Update model parameter θ with gradient under learning rate η

$\log\vec{a}_i \leftarrow \text{LogSoftmax}\log\vec{a}_i$

end

Save the best estimated model $\hat{\theta} = \theta_{\text{best}}$ on the validate dataset with $\min\sum_{b=1}^{\mathrm{N}_{\text{valid}}}(\hat{y}_b - y_b)^2$.

Testing:

Predict the testing data set with Algorithm S1 with parameter $\hat{\theta}$.

the gas chromatography for carbon-dioxide concentration measurement has large time delay.

To deal with measuring problem, and improve the control quality of the carbon-dioxide absorber, soft sensor models have been adopted to estimate the carbon-dioxide concentration at the outlet stream of the absorber in real time. To this end, several hard sensors are installed on the plant to collect the samples for process variables, which can be used as secondary variables for soft sensor. Eleven process variables marked in red zone are selected to construct the model. Table SI gives the detailed description of these variables. A total of 6, 000 samples are collected from the process.

TABLE SI
PROCESS VARIABLES FOR CAC

Input Variables	Description
U1	Pressure of inlet gas
U2	Liquid level of buffer vessel
U3	Temperature of inlet barren liquor
U4	Flowrate of inlet lean solution
U5	Flowrate of inlet semi-lean solution
U6	Temperature of inlet gas
U7	Pressure drop of absorber
U8	Temperature of rich solution
U9	Liquid level of absorber
U10	Liquid level of the separator
U11	Pressure of outlet gas
Y	Outlet carbon-dioxide concentration

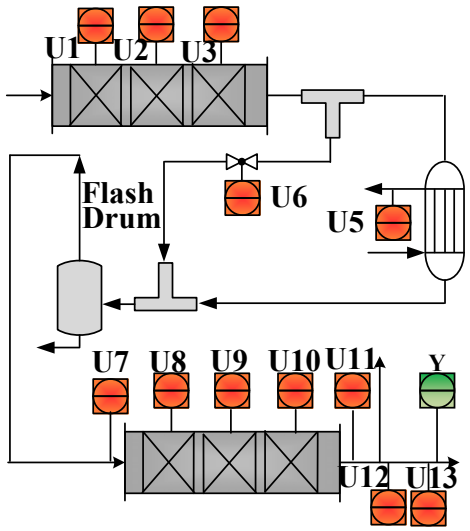


Fig. S2. Flowsheet of CSC

B. Catalysis Shift Conversion Unit

Fig. S2 presents the flowsheet of CSC Unit in an ammonia synthesis process [SR9]. The following heterogeneous catalytic reaction will take place in the fixed-bed reactors connected in series:

$$\mathrm{CO} + \mathrm{H_2O} \leftrightarrow \mathrm{CO_2} + \mathrm{H_2}. \tag{S27}$$

To estimate the carbon-monoxide concentration marked in the green zone in real-time for the sake of meeting the technology requirement of carbon hydrogen ratio, several hard sensors are installed on the section to collect process variables in real time. Thirteen covariates marked in the red zone in Fig. S2 are chosen for soft sensor modeling, and the detailed descriptions are given in Table SII.

C. Methanation Furance Unit

Fig. S3 presents the flowsheet of MFU. In the MFU, the gas from CAC will be fed to the heat exchanger connected in series to decrease the gas temperature for the sake of increasing conversion. Based on this, the reaction gas is fed to the fixed-bed-reactor and go through the reaction given in (S28):

$$\begin{cases} \mathrm{CO} + 3\mathrm{H_2} \rightarrow \mathrm{CH_4} + \mathrm{H_2O} \\ \mathrm{CO_2} + 4\mathrm{H_2} \rightarrow \mathrm{CH_4} + 2\mathrm{H_2O} \end{cases}. \tag{S28}$$

TABLE SII
THE PROCESS VARIABLES IN THE CSC PROCESS

Process variables	Description
U1	High temperature bed temperature 1
U2	High temperature bed temperature 2
U3	High temperature bed temperature 3
U4	Outlet temperature of high temperature bed
U5	Outlet temperature of cooling water
U6	Split-gas temperature
U7	Inlet temperature of low temperature bed
U8	Low temperature bed temperature 1
U9	Low temperature bed temperature 2
U10	Low temperature bed temperature 3
U11	Outlet temperature of low temperature bed
U12	Outlet pressure of low temperature bed
U13	Product gas pressure
Y	Outlet carbon-monoxide concentration

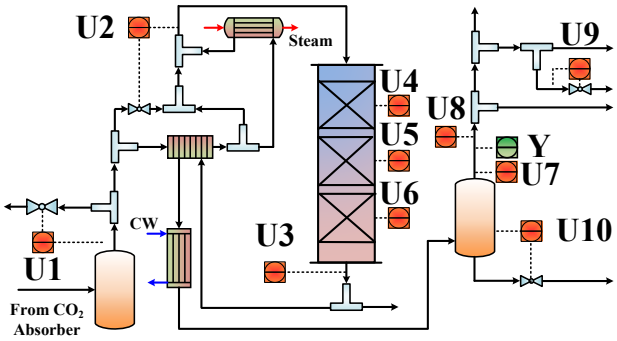


Fig. S3. The flowsheet of MFU.

The methanation reaction, a key process in CO and CO_2 to methane, is notably exothermic. Specifically, converting each 1% of CO can lead to a temperature increase of approximately 74°C, while each 1% of CO_2 conversion results in a 60°C rise in temperature. Consequently, an increase in the CO and CO_2 content within the process gas significantly elevates the reaction temperature. To manage this, it is generally recommended to reduce the inlet temperature to control the reactor's outlet temperature effectively. The typical operating temperature of a methanation reactor is about 30°C. To monitor the conversion rate in real-time, based on the concentrations of CO and CO_2, we select ten process variables. These variables include flows, pressures, temperatures, and liquid levels, highlighted in red in Fig. S3, to develop an industrial data-driven model. The concentrations of CO and CO_2 are estimated using gas chromatography, illustrated in green in Fig. S3. Detailed information about these variables is provided in Table SIII. This approach ensures precise control and optimization of the methanation process, enhancing both efficiency and safety. In total, 3,000 samples were collected from the process.

D. Primary Reformer Unit

Fig. S4 depicts the flowsheet of the PRU as described in [SR10]. In this technical design, natural gas (NG) and high-pressure steam are fed into fixed tube array reactors arranged in parallel. The reactions, as outlined in (S29), facilitate the production of hydrogen for subsequent ammonia synthesis. Among the technical specifications, reaction temperature is

TABLE SIII
PROCESS VARIABLES IN THE MFU

Variables	Descriptions
U1	Flowrate of upstream gas
U2	Gas pressure for reactor entrance
U3	Gas Pressure for reactor outlet
U4	Temperature of upper fix-bed
U5	Temperature of middle fix-bed
U6	Temperature of lower fix-bed
U7	Pressure of flash drum
U8	Flowrate of outlet drum
U9	Pressure of purge gas
U10	Flowrate of outlet waste water
Y	Content of CO and CO_2

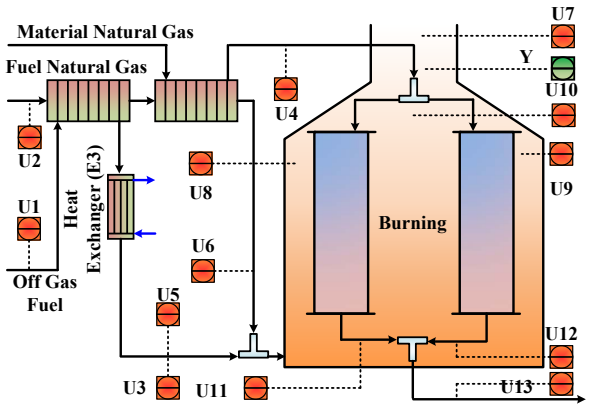


Fig. S4. The flowsheet of PRU.

paramount for optimizing hydrogen yield. The relevant reactions are:

$$\begin{cases} C_nH_{2n+2} + nH_2O \overset{\text{Heat}}{\leftrightarrow} nCO_2 + (2n+1)H_2 \\ CH_4 + H_2O \overset{\text{Heat}}{\leftrightarrow} CO + 3H_2 \\ CO + H_2O \overset{\text{Heat}}{\leftrightarrow} CO_2 + H_2 \end{cases}, \quad \text{(S29)}$$

where C_nH_{2n+2} indicates alkane, CH_4 is methane, and CO is carbon monoxide.

Additionally, the process necessitates real-time monitoring of oxygen content at the outlet of the low-temperature bed, indicated in green in Fig. S4. Traditional measurement via mass spectrometry leads to significant delays. To address this, thirteen variables have been selected for industrial data-driven modeling, with detailed descriptions provided in Table SIV. In total, 3,000 samples were collected from the process.

TABLE SIV
PROCESS VARIABLES IN THE PRU

Variables	Description
U1	Flow rate of fuel NG
U2	Flow rate of fuel off gas
U3	Pressure of fuel off gas at E3's exit
U4	Pressure of flue gas at PR's exit
U5	Temperature of fuel off gas at E3's exit
U6	Temperature of fuel NG at preheater's exit
U7	Temperature of feed gas at PR's entrance
U8	Temperature of flue gas at PR's top left
U9	Temperature of flue gas at PR's top right
U10	Temperature of flue gas at PR's top
U11	Temperature of production gas at PR's left
U12	Temperature of production gas at PR's right
U13	Temperature of production gas at PR's exit
Y	Outlet Oxygen Content

E. Prior Knowledge Analysis

1) Prior Knowledge Description and Graph Formation: Unlike fields such as Quality-of-Service data analysis [SR11] and recommender systems [SR12], where the graph structure is readily available and can be directly utilized in model development, the context of industrial soft sensor modeling, which is the focus of this work, presents different challenges. The data collected from process instrumentation variables typically assume a tabular form.

Given this, to evaluate non-adaptive GNNs such as graph convolutional networks and graph attention networks, we propose using the mean-field assumption for the graph. We treat the edges of the graph as following a Bernoulli distribution [SR13, SR14] and perform sampling from this 'prior graph' to form the matrix $\mathcal{A}$, which is then used to construct and evaluate the model on this sampled graph. To this end, we introduce a description approach for industrial process variables: Based on our understanding of the industrial process, we suggest the incorporation of logical parameters $\boldsymbol{s}_i|_{i=1}^3$ to delineate the relationship between industrial process knowledge and model parameters:

- $\boldsymbol{s}_1$: there exist prior knowledge between the process variables i and j.
- $\boldsymbol{s}_2$: the existence of prior knowledge about the process variables i and j is unknown.
- $\boldsymbol{s}_3$: there exists no prior knowledge between the process variables i and j.

Building on this framework, we adopt the $2-\sigma$ bound from the Gaussian distribution according to the Pauta criterion for the sake of determining the value of ρ, which represents the 'strength' of the edge connections in the prior graph. Consequently, the following disjunctive expression can be formulated:

$$\left[\begin{array}{c}\boldsymbol{s}_1\\ \rho=0.955\end{array}\right]\veebar\left[\begin{array}{c}\boldsymbol{s}_2\\ \rho=0.500\end{array}\right]\veebar\left[\begin{array}{c}\boldsymbol{s}_3\\ \rho=0.045\end{array}\right], \tag{S30}$$

where $\veebar$ stands for disjunction symbol, and (S30) indicates that only one square brackets will take effect. Based on this, we can formulate the 'prior graph' ϱ, which is consists of ρ.

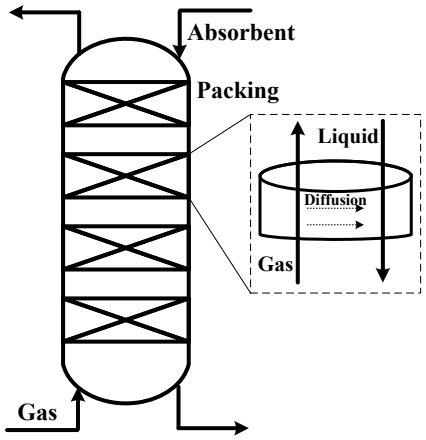


Fig. S5. The absorber model.

2) Prior Knowledge Analysis of Absorber: Fig. S5 illustrates the functional design of an absorber, a critical component in chemical processing. The absorber is fundamentally composed of various layers of packing material stacked in series. During operation, gas enters from the bottom of the column and is directed upward, while the absorbent liquid is introduced from the top and flows downward. The packing within the absorber plays a crucial role in disrupting the flow of the absorbent, effectively increasing the contact area between the gas and the liquid. This design maximizes the interaction between the ascending gas and the descending liquid, facilitating the diffusion of solutes from the gas phase into the liquid phase based on their solubility in the absorbent. This process effectively strips undesirable components from the gas stream, capturing them in the liquid absorbent. On this basis, the concentration of the outlet gas can be calculated according to the short-cut equations [SR15] as follows:

$$\begin{cases}\mathcal{F}_i^{\text{absorb}}=\dfrac{\mathcal{L}_{\text{absorb}}}{\mathcal{V}_{\text{gas}}\mathcal{K}_i}\\ \varphi_i=\dfrac{\left(\mathcal{F}_i^{\text{absorb}}\right)^{\text{N}_{\text{tray}}+1}-\mathcal{F}_i^{\text{absorb}}}{\left(\mathcal{F}_i^{\text{absorb}}\right)^{\text{N}_{\text{tray}}+1}-1}\\ y_i^{\text{out}}=(1-\varphi_i)\times y_i^{\text{in}}\end{cases} \tag{S31}$$

where i is the category of components to be absorbed in the gas, $\mathcal{F}_i^{\text{absorb}}$ is the absorption factor, $\mathcal{K}_i$ is the phase equilibrium, $\mathcal{L}_{\text{absorb}}$ is the flow rate of the absorbent, $\mathcal{V}_{\text{gas}}$ is the flowrate of the gas, y_i^{in} is the concentration of component i in the inlet gas, y_i^{out} is the concentration of component i in the outlet gas, and N_{tray} is the tray number of absorber.

By analyzing the flowsheet given in Fig. S1, the flow rate of inlet gas can be equivalent to the inlet gas pressure. The flow rate of lean and semi-lean absorbent can be equivalent to the solution flowrate. Besides, in the absorption process, the temperature of outlet rich solution determined by the inlet solution flowrate can reflect the absorptivity from the perspective of chemical reaction kinetics and thermodynamics. Therefore, the prior knowledge of the process can be stated as follows:

- U4 and U5 should connect with U1
- U8 should connect with U4 and U5
- U1 may not connect to other vaules

Based on the abovementioned items, the prior knowledge is plotted in Fig. S7 (a) as matrix ϱ.

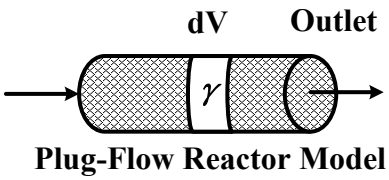
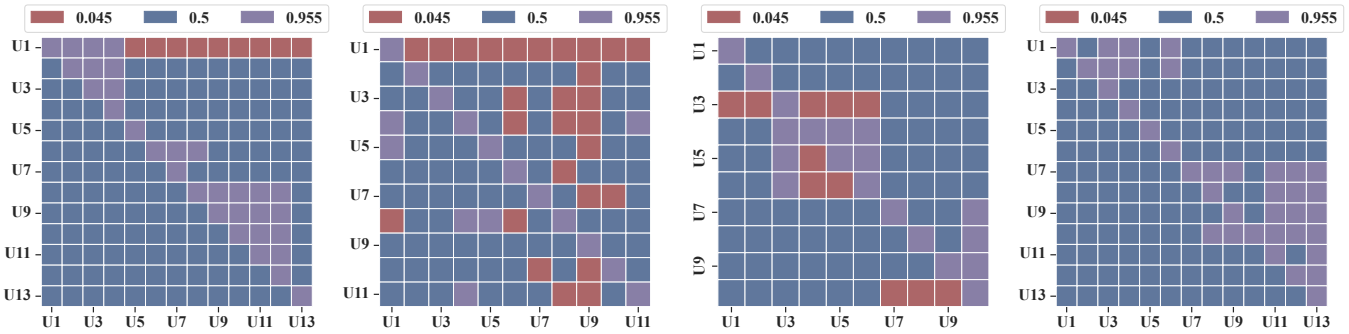


Fig. S6. The plug-flow reactor model.

3) Prior Knowledge Anaylsis of Reactor: According to chemical reaction engineering, the reactors are analyzed using the plug-flow reactor model. in Fig. S6. By analyzing the infinitesimal marked in the white zone, the mass conservation of reactant γ can be derived as (S32) shown, where the ζ is the bed voidage, F is the flowrate, R is the reaction rate.



(a) Prior knowledge of CSC. (b) Prior knowledge of CAC. (c) Prior knowledge of MFU. (d) Prior knowledge of PRU.

Fig. S7. The heat map of prior knowledge to form matrix ϱ.

$$F_\gamma - (F_\gamma + \mathrm{d}F_\gamma) = (-R_\gamma)(1-\zeta)\gamma_i, \tag{S32}$$

Meanwhile, the heat conservation [SR16] can be given in (S33) shown.

$$Gc_pT - Gc_p(T+\mathrm{d}T) = R_\gamma(1-\varsigma_B)(-\Delta H)\gamma_i \mathrm{d}l, \tag{S33}$$

According to Ergun Equation, the pressure drop can be derived as follows:

$$\frac{\mathrm{d}p}{\mathrm{d}l} = (\frac{150}{\mathrm{Re}} + 1.75)(\frac{1-\eta}{\zeta^3})(\frac{\rho_g u_m}{d_s}), \tag{S34}$$

where the Re is Reynold's number, ρ_g is the reactant gas density, and u_m is the gas velocity. Note that the gas velocity determines the ratio of gas reactant diffusion rate and reaction rate (also known as Thiele modulus). And thus, in the designing stage of the reactor, the velocity and the voidage should be well designed to make the pressure drop as small as possible for the purpose of higher reactant conversion [SR17].

On this basis, the following prior knowledge can be obtained as follows for CSC according to Fig. S2:

- U1, U2 can derive the temperature of U3, U4 and U6, and may influence the burning temperature.
- Temperature sensors U7, U10 and temperature sensors U8, U9, U11, U12, and U13 can derive the temperature difference
- Temperature sensors U8, U9 and temperature sensors U11, U12, and U13 can derive the temperature difference.
- Temperature sensors U11, U12 and temperature sensors U13 can derive the temperature difference.

Based on the abovementioned items, the prior knowledge is plotted in Fig. S7 (b) as matrix ϱ.

For MFU process according to Fig. S3, we can have following knowledge:

- Temperature sensors U4 and temperature sensors U2, U3, and U5 can derive the temperature difference.
- Temperature sensors U5 and flow rate sensor U2 can derive the temperature difference.
- Sensors U7, U8, U9 and sensor U10 can derive the temperature/ pressure difference.

Based on the abovementioned items, the prior knowledge is plotted in Fig. S7 (c) as matrix ϱ.

For PRU process according to Fig. S4, we can have following knowledge:

- U1, U2 can derive the temperature of U3, U4 and U6, and may influence the burning temperature.
- Temperature sensors U7, U10 and temperature sensors U8, U9, U11, U12, and U13 can derive the temperature difference.
- Temperature sensors U8, U9 and temperature sensors U11, U12, and U13 can derive the temperature difference.
- Temperature sensors U11, U12 and temperature sensors U13 can derive the temperature difference.

Based on the abovementioned items, the prior knowledge is plotted in Fig. S7 (d) as matrix ϱ.

F. Dataset Statistics & Utilization Guidelines

Based on previous subsection, we further propose the dataset statistic results as follows:

TABLE SV
DATASET STATISTICS

Dataset Name	Dataset Size	Node No.	ϱ for edge sampling
CSC	7,000	13	Fig. S7 (a)
CAC	6,000	11	Fig. S7 (b)
MFU	2,000	10	Fig. S7 (c)
PRU	3,000	13	Fig. S7 (d)

Based on this foundation, we provide the following guidelines for applying the proposed E^{2}AG algorithm to real industrial processes, as detailed in Algorithm Algorithms S1 and S2:

1) **Hyperparameter Determination**: Determine key hyperparameters such as the number of heads H, model batch size $\mathcal{B}$, and learning rate η. Proceed with model training as specified in Algorithm Algorithms S1 and S2.
2) **Data Collection & Model Training**: Sequentially collect covariates $[x_1, ..., x_\mathrm{D}]$ and corresponding labels y to formulate the training dataset and facilitate model training.
3) **Rolling Testing**: Continuously test the model along the time axis in a rolling fashion by repeating Step 2).

S.III. BASELINE MODELS & CONCERNING HYPERPARAMETER SETTINGS

A. Baseline Models & Reasons

To provide a thorough comparison and showcase the effectiveness of E^{2}AG model, the baseline models categorized as follows are chosen for prediction accuracy comparison:

- **MHSAM-based Models:** Flashformer [SR18], iTransformer [SR19], Random Synthesizer (Synthesizer (Random)) [SR20], and Dense Synthesizer (Synthesizer (Dense)) [SR20].
- **AGNN-based Models:** Deep Learning model with Dynamic Graph (DGDL) [SR21] and stacked graph convolutional network (S-GCN) [SR22]
- **Fix Graph Models:** Two-Stream Graph Convolutional Network-Incorporated Latent Feature Analysis (TGLFA) [SR11], Symmetry-Preserving Dual-Stream Graph Neural Networks (SDGNN) [SR23], Adaptive Graph Contrastive Learning (AdaGCL) [SR12], Graph Convolution Network with Random Weight (GCN-RW) [SR24], and UniFilter [SR25].

Notably, *the iTransformer is the state-of-the-art model published in 'The Twelfth International Conference on Learning Representations' (ICLR-2024) in the time-series forecasting task.* As outlined above, the proposed E^{2}AG model draws inspiration from MHSAM. Therefore, Transformer models recently presented at various conferences have been selected as baseline models for comparison. Furthermore, to our knowledge, approaches for constructing adaptive graphs can be categorized as follows: 1) treating the graph as a model parameter, and 2) predicting the graph through an auxiliary NN trained jointly with the main model. Consequently, within the realm of AGNN-based models, Synthesizer (Random), Synthesizer (Dense), and S-GCN have been chosen as representative baselines. It is worth noting that the graphs in Synthesizer (Random) and S-GCN are learnable parameters, while the graph in Synthesizer (Dense) is generated by an auxiliary NN. Furthermore, to illustrate the comprehensive capabilities of the proposed E^{2}AG model, we have also included other GNNs with predefined fixed graphs as baseline models. These models have achieved significant success in fields such as Quality of Service (QoS) data analysis and recommender systems. Notable examples include TGLFA, SDGNN, AdaGCL, among others.

B. Hyperparameter Selection

Through a comprehensive grid search, the hyperparameters for all models have been tabulated in Table SVI. The construction of fixed graph models for inference is based on Bernoulli sampling applied to the prior graph, as detailed in Section S.II.E1. In addition, for fairness, the training epoch for all models is set as 200.

TABLE SVI
HYPERPARAMETER TABLE

Model	Dataset	CSC					CAC				
	Hyperparameters	$\mathcal{B}$	η	d^{embed}	N_{hop}	H	$\mathcal{B}$	η	d^{embed}	N_{hop}	H
Flashformer		128	0.02	16	1	2	128	0.02	16	1	2
iTransformer		128	0.02	16	1	2	128	0.01	16	1	2
Synthesizer (Random)		128	0.02	16	1	2	128	0.02	16	1	2
Synthesizer (Dense)		512	0.01	16	1	2	512	0.01	16	1	2
S-GCN		32	0.001	16	3	-	32	0.001	16	3	-
TGLFA		128	0.01	32	2	-	128	0.01	32	2	-
SDGNN		64	0.001	16	2	-	64	0.001	16	2	-
AdaGCL		64	0.01	32	2	-	64	0.001	16	2	-
GNNRW		-	-	-	1	-	-	-	-	1	-
UniFilter		64	0.01	16	3	-	64	0.01	16	3	-
DGDL		128	0.001	32	2	-	256	0.001	32	2	-
E^{2}AG		128	0.02	1	1	6	128	0.02	1	1	6
Model	Dataset	MFU					PRU				
	Hyperparameters	$\mathcal{B}$	η	d^{embed}	N_{hop}	H	$\mathcal{B}$	η	d^{embed}	N_{hop}	H
Flashformer		128	0.001	32	1	8	128	0.02	16	1	2
iTransformer		64	0.001	32	1	8	64	0.001	32	1	8
Synthesizer (Random)		128	0.001	32	1	8	128	0.001	32	1	8
Synthesizer (Dense)		128	0.001	32	1	8	128	0.001	32	1	8
S-GCN		64	0.001	32	3	-	64	0.001	32	3	-
TGLFA		128	0.01	32	2	-	128	0.01	32	2	-
SDGNN		64	0.001	16	2	-	64	0.001	16	2	-
GCLAda		128	0.01	16	2	-	512	0.01	16	2	-
GNNRW		-	-	-	1	-	-	-	-	1	-
UniFilter		64	0.01	32	3	-	64	0.01	32	3	-
DGDL		32	0.001	32	2	-	32	0.001	32	2	-
E^{2}AG		128	0.01	1	1	6	64	0.01	1	1	6

REFERENCES

[SR1] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," *Advances in neural information processing systems*, vol. 29, 2016.

[SR2] D. Wang and Q. Liu, "Nonlinear stein variational gradient descent for learning diversified mixture models," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6576–6585.

[SR3] H. Dong, X. Wang, L. Yong, and T. Zhang, "Particle-based variational inference with preconditioned functional gradient flow," in *The Eleventh International Conference on Learning Representations*, 2022, pp. 1–26.

[SR4] D.-X. Zhou, "Derivative reproducing properties for kernel methods in learning theory," *Journal of computational and Applied Mathematics*, vol. 220, no. 1-2, pp. 456–463, 2008.

[SR5] J. Shi, S. Sun, and J. Zhu, "A spectral approach to gradient estimation for implicit distributions," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4644–4653.

[SR6] F. Santambrogio, "{Euclidean, metric, and Wasserstein} gradient flows: an overview," *Bulletin of Mathematical Sciences*, vol. 7, pp. 87–154, 2017.

[SR7] J. C. Butcher, *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[SR8] B. Shen, L. Yao, and Z. Ge, "Nonlinear probabilistic latent variable regression models for soft sensor application: From shallow to deep structure," *Control Engineering Practice*, vol. 94, p. 104198, 2020.

[SR9] W. Shao, Z. Ge, and Z. Song, "Semisupervised bayesian gaussian mixture models for non-gaussian soft sensor," *IEEE Transactions on Cybernetics*, vol. 51, no. 7, pp. 3455–3468, 2021.

[SR10] L. Yao and Z. Ge, "Cooperative deep dynamic feature extraction and variable time-delay estimation for industrial quality prediction," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, 2021.

[SR11] F. Bi, T. He, Y. Xie, and X. Luo, "Two-stream graph convolutional network-incorporated latent feature analysis," *IEEE Transactions on Services Computing*, vol. 16, no. 4, pp. 3027–3042, 2023.

[SR12] Y. Jiang, C. Huang, and L. Huang, "Adaptive graph contrastive learning for recommendation," in *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 2023, pp. 4252–4261.

[SR13] P. Elinas, E. V. Bonilla, and L. Tiao, "Variational inference for graph convolutional networks in the absence of graph data and adversarial settings," *Advances in neural information processing systems*, vol. 33, pp. 18 648–18 660, 2020.

[SR14] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, vol. 32, pp. 1–12, 2019.

[SR15] J. D. Seader, E. J. Henley, and D. K. Roper, *Separation process principles*. Wiley New York, 1998.

[SR16] W. L. McCabe, J. C. Smith, and P. Harriott, *Unit operations of chemical engineering*. McGraw-hill New York, 1993, vol. 5.

[SR17] J. R. Ross, "Chapter 8 - mass and heat transfer limitations and other aspects of the use of large-scale catalytic reactors," in *Contemporary Catalysis*, J. R. Ross, Ed. Amsterdam: Elsevier, 2019, pp. 187–213.

[SR18] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems (NeurIPS 2022)*, 2022, pp. 1–16.

[SR19] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "iTransformer: Inverted transformers are effective for time series forecasting," in *The Twelfth International Conference on Learning Representations*, 2024, pp. 1–22.

[SR20] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, and C. Zheng, "Synthesizer: Rethinking self-attention for transformer models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 183–10 192.

[SR21] K. Zhu and C. Zhao, "Dynamic graph-based adaptive learning for online industrial soft sensor with mutable spatial coupling relations," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 9, pp. 9614–9622, 2023.

[SR22] Y. Wang, Q. Sui, C. Liu, K. Wang, X. Yuan, and G. Dong, "Interpretable prediction modeling for froth flotation via stacked graph convolutional network," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 1, pp. 334–345, 2024.

[SR23] J. Chen, Y. Yuan, and X. Luo, "SDGNN: Symmetry-preserving dual-stream graph neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 7, pp. 1717–1719, 2024.

[SR24] C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, and X. Wu, "Are graph convolutional networks with random weights feasible?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 2751–2768, 2023.

[SR25] K. Huang, Y. G. Wang, M. Li, and P. Lio, "How universal polynomial bases enhance spectral graph neural networks: Heterophily, over-smoothing, and over-squashing," in *Proceedings of the 41st International Conference on Machine Learning (ICML' 24)*, ser. Proceedings of Machine Learning Research, vol. 235. PMLR, 21–27 Jul 2024, pp. 20 310–20 330.